МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Погорецький Віталій Миколайович

Програмні засоби нейромережевої класифікації біомедичних зображень / Software of neural network classification of biomedical images

спеціальність: 6.050102 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерні системи та мережі

Випускна кваліфікаційна робота

Виконав: студент групи КСМ-42/1
Погорецький Віталій Миколайович
_____

Науковий керівник:
Піцун О. Й.
_____

Випускну кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
О. М. Березький

ТЕРНОПІЛЬ - 2019

РЕЗЮМЕ

Дипломний проект містить 82 сторінки пояснюючої записки, 38 рисунків, 10 таблиць, 3 додатки. Обсяг графічного матеріалу 2 аркуші формату А3.

Метою дипломного проекту є розробка програмного засобу для класифікації біомедичних зображень, використовуючи нейронну мережу.

В дипломному проекті на основі порівняльного аналізу трьох популярних класифікаторів зображень виділено основні тези при їх використанні та результати класифікації, що дозволило сформувати ясне бачення різниці в роботі класифікаторів.

Враховуючи ріст популярності застосування інформаційних технологій в освіті, розроблено клієнт-серверну систему класифікації гістологічних та цитологічних зображень. Клієнтська частина додатку виконана з використанням популярної бібліотеки React.js. Серверна частина написана на мові програмування PHP з використанням бібліотеки PHP-ML для маніпулювання штучним інтелектом.

Користувацький інтерфейс забезпечує завантаження файлів для тренування та прогнозування і вибір одного з трьох методів класифікації, після чого користувач отримає інформацію про точність правильно відгаданих класів зображень.

Можливість саме такого зручного аналізу методів класифікації дозволить знайти кращого з них при тих чи інших даних, а також значно підвищить продуктивність роботи медичного закладу.

Ключові слова: ВЕБ-ДОДАТОК, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, КЛІЄНТ-СЕРВЕРНИЙ-ДОДАТОК, МЕТОД КЛАСИФІКАЦІЇ, БІОМЕДИЧНЕ ЗОБРАЖЕННЯ.

RESUME

The degree project contains 82 pages of explanatory note, 38 figures, 10 tables, 3 annexes. The volume of graphic material is 2 sheets of A3 format.

The aim of the degree project is the development of software for the classification of biomedical images using a neural network.

In the degree project, on the basis of a comparative analysis of three popular image classifiers, the main theses for their use and the results of the classification are highlighted, which allowed to form a clear vision of the difference in the classifiers.

Taking into account the growing popularity of the use of information technologies in education, a client-server classification system for histological and cytological images has been developed. The client part of the application is made using the popular React.js library. The server part is written in the PHP programming language using the PHP-ML library for manipulating artificial intelligence.

The user interface provides download files for training and forecasting and the choice of one of the three classification methods, after which the user will receive information about the accuracy of correctly guessed image classes.

The possibility of just such a convenient analysis of classification methods will allow you to find the best of them with certain data, and also significantly increase the productivity of the medical institution.

Keywords: WEB APPLICATION, IMAGE CLASSIFICATION, CLIENT SERVER APPLICATION, CLASSIFICATION METHOD, BIOMEDICAL IMAGE.

ЗМІСТ

ВСТУП

Останнім часом одним з актуальних напрямків розвитку комп'ютерних технологій в медицині стає обробка цифрових зображень: покращення якості зображення, відновлення пошкоджених зображень, розпізнавання окремих елементів. Розпізнавання патологічних процесів є однією з найбільш важливих завдань обробки та аналізу медичних зображень.

Швидкий розвиток сучасної медицини супроводжується тісною взаємодією з суміжними областями – математикою, фізикою, хімією. Зображення анатомічної, гістологічної будови і функцій людського тіла є фундаментальними для медичної науки. Діагностика захворювань, лікування і управління терапевтичними процедурами спираються на дані, одержувані медичною візуалізацією.

Істотна варіабельність і слабка контрастність біологічних і медичних зображень є основними труднощами в задачах вимірювання і розпізнавання. Для їх подолання потрібна професійна інтуїція дослідника і виконання складної і рутинної роботи з коригування результатів. Застосування обчислювальної техніки значно спрощує вирішення цих завдань, полегшуючи роботу з медичними зображеннями.

В наш час у медицині відбувається поглиблене вивчення розвитку патологічних змін в організмі людини, тому виникає потреба в створенні сучасних інформаційних пристроїв і методів обробки біомедичної інформації, зокрема зображень. Відомо, що при встановленні діагнозу і проведенні лікування лікарі все частіше покладаються на біомедичні зображення, отримані з використанням різних апаратних і програмних комплексів. Зокрема, це стосується автоматичної обробки томографічних і термографічних зображень для сегментації зображень і завдань класифікації.

Коли алгоритми виконують розпізнавання на рівні людини-експерта, автоматизація веде до прискорення роботи систем обробки даних і підвищенню

їх ефективності. Знаходження «аномальних» клітин на медичному зображенні є досить трудомістким завданням. Тому бажано мати засіб автоматичного розпізнавання, такий, щоб медичні експерти розглядали тільки складні випадки.

Основною метою даного дипломного проекту є розробка нейромережі для класифікації біомедичних зображень.

Актуальність розробки даної нейронної мережі підтверджується величезною кількістю найрізноманітніших практичних застосувань в області медицини і не тільки.

1 АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

1.1 Класифікація біомедичних зображень

Зображення служить для подання інформації у візуальному вигляді. Воно є однією з найбільш зручних форм представлення інформації при діагностуванні органів людини в медицині.

Біомедичними зображеннями будемо називати зображення, отримані за допомогою будь-якої біомедичної техніки, що використовуються для візуального та автоматизованого аналізу у медицині та біології. Варто відзначити, що біомедичне зображення – відносно нове поняття в променевій діагностиці.

Основними способами отримання біомедичних зображень є променеві методи дослідження, а саме: рентгенологічний, магнітно-резонансний, радіонуклідний та ультразвуковий.

Класифікацію біомедичних зображень наведено у таблиці 1.1.

Таблиця 1.1 – Класифікація біомедичних зображень

Критерій класифікації		
За способом отримання зображення	За типом зображення	За розмірністю
цифрова радіологія; комп'ютерна томографія; ядерний магнітний резонанс; ультразвук; мікроскопія.	RGB зображення (мікроскопія, термографія); зображення в градаціях сірого (томографія); чорно-біле (ультразвук).	2D (всі зображення незалежно від способу отримання); 3D (послідовність радіологічних зображень, томографічне зображення).

Умовно можна виділити наступні критерії класифікації біомедичних зображень: за способом отримання зображення, за типом зображення та за розмірністю зображення.

Комп'ютерна томографія (КТ) – це метод пошарової діагностики організму, заснований на рентгенівському випромінюванні. Але якщо при звичайному рентгені промені проходять крізь тіло і фокусуються на плівці або пластині, даючи двомірне зображення, то при виконанні КТ зображення виходить об'ємним. Справа в пристрої апарату для КТ: джерелом рентгенівських променів служить кільцеподібний контур, усередині якого розташована спеціальна кушетка (стіл) для пацієнта. Таким чином виконується ціла серія рентгенівських знімків органів, отриманих з різних точок і під різним кутом. За допомогою комп'ютера все зображення обробляються, і в підсумку моделюється тривимірне зображення органу. Важливо, що лікар має можливість подивитися «зрізи» органу: в залежності від налаштувань апарату, товщина зрізу може становити до 1 мм. Це збільшує точність діагностики [1]. Сучасні комп'ютерні томографи дозволяють отримувати зображення з високою просторовою роздільною здатністю за короткий проміжок часу.

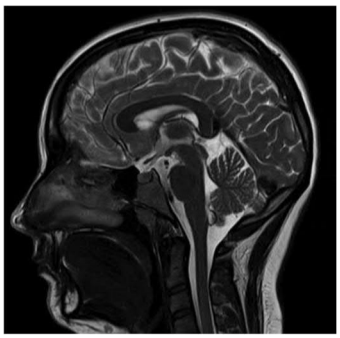
Магнітно-резонансна томографія (МРТ) заснована на тому ж принципі, що і КТ: отримання масиву даних і моделювання на його основі тривимірного зображення органу. Різниця з КТ полягає в природі хвиль: при МРТ вони електромагнітні. Під їх дією різні ділянки тканин дають різну «відповідь», яка фіксується приймальним пристроєм апарату. А потім, точно так же, як і при КТ, сигнали обробляються і перетворюються в зображення. Результатом МРТ є 2D або 3D зображення в градаціях сірого. Перевага використання 2D МРТ зображення є вища швидкість формування зображень у порівнянні з 3D, а також 2D зображення краще підходять для дослідження довгих судин. Перевагою використання 3D зображень є висока просторова роздільна здатність у порівнянні з 2D та відображення об’єкту у тривимірному просторі [2].

Ультразвукова діагностика (УЗД) – це метод візуалізації на основі використання високочастотних звукових хвиль для отримання поперечних

зображень тіла [3]. Принцип роботи УЗД заснований на тому, що ультразвукові хвилі проходять крізь тканини різної щільності і по-різному відбиваються від них, передаючи інформацію спеціальному приймає датчику, який фіксує дані і переводить їх в графічне зображення, видаючи його на моніторі або спеціальному фотопапері [4].

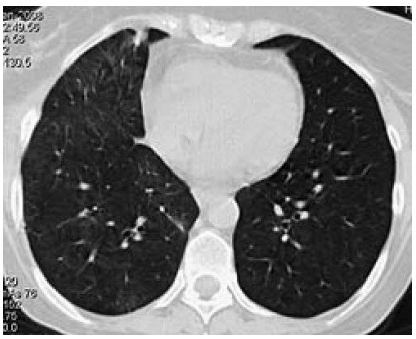
Гістологічні та цитологічні зображення – це RGB зображення, що формуються в результаті мікроскопічного дослідження. Гістологія вивчає будову організму на тканинному рівні. Цитологія по суті є одним з підрозділів гістології – вивчення клітин живих організмів [5].

Спрощено можна сказати, що коли береться біологічна тканина для досліджень, то спочатку проводиться гістологічне дослідження – вивчення структури тканини різними методами,а потім тканина досліджується на більш глибокому рівні – на клітинному, тобто робиться цитологічне дослідження. Відповідно в цитологічних зображеннях навідміну від гістологічних, клітини знаходяться поза межами тканини [6].

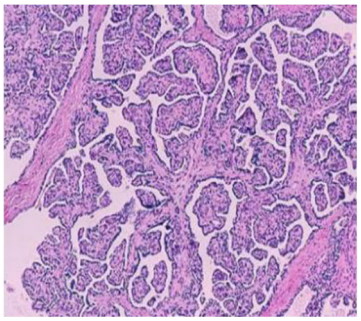


а) МРТ зображення

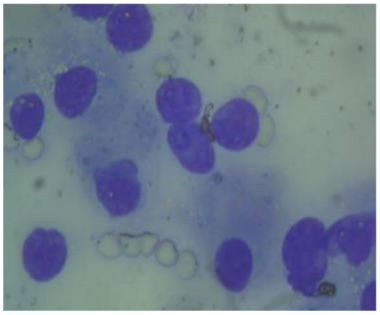
б) УЗД зображення



в) КТ зображення



Д) гістологічне зображення



ж) цитологічне зображення

Рисунок 1.1 – Приклади медичних зображень

На рисунку 1.1 зображено візуальне порівняння медичних зображень різних методів дослідження, тим самим чітко різницю між ними та з чим вони працюють.

1.2 Аналіз засобів класифікації зображень

Класифікація об'єктів є легким завданням, але вона складна для машини. Класифікація зображень включає в себе попередню обробку зображень, датчики зображень, виявлення об'єктів, сегментацію об'єктів, витягання ознак і класифікацію об'єктів. Система класифікації зображень складається з бази даних, яка містить визначені наперед шаблони, які порівнюються з об'єктом для класифікації в відповідну категорію. Класифікація зображень є важливим завданням в різних областях, таких як дистанційне зондування, біометрія, біомедичні зображення і навігація роботів. Типова система класифікації складається з камери, встановленої високо в зацікавленій зоні, де зображення захоплюються і, отже, обробляються.

Класифікація зображень включає в себе наступні етапи:

– отримання зображень для обробки зображень;

– попередня обробка зображень, використовуючи методи видалення шуму, атмосферної корекції і т.п;

– витягання важливих характеристик зображення.

Зображення класифікуються на основі витягнутих елементів в попередньо визначені категорії з використанням відповідних методів, які порівнюють шаблон зображення з зображеннями, які знаходяться всередині бази даних.

Можна виділити такі основні методи класифікації: метод опорних векторів, логістична регресія та нейронні мережі. Розлянемо кожен з них більш детально, але перед цим необхідно зрозуміти декілька понять.

Класифікація – віднесення об'єкта до однієї з категорій на підставі його ознак.

Регресія – прогнозування кількісної ознаки об'єкта на підставі інших його ознак.

Кластеризація – розбиття множини об'єктів на групи на підставі ознак цих об'єктів так, щоб усередині груп об'єкти були схожі між собою, а поза однієї групи – менш схожі.

Бінарний класифікатор – класифікатор, що оцінює до якого з двох можливих класів (наприклад «норма» і «патологія») слід віднести досліджуваний об'єкт [7].

Нейронна мережа (рисунок 1.2) являє собою структуру, що складається з вузлів – штучних нейронів, які найчастіше пов'язані пошарово таким чином, щоб якийсь сигнал (вектор або масив, розмірність якого дорівнює числу вхідних нейронів), що подається на вхідний шар, був перетворений у вихідний шар, який являє собою рішення задачі. Кожен зв'язок має вагу, а вузол – функцію активації [8].

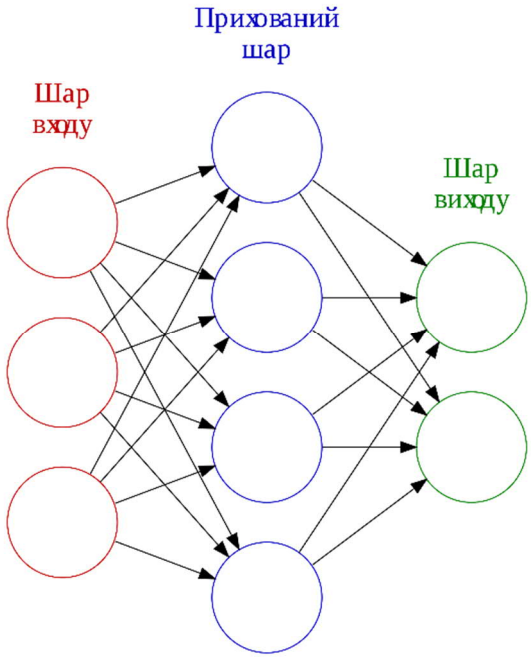


Рисунок 1.2 – Приклад нейронної мережі

Семантична мережа (рисунок 1.3) є одним з різновидів мережевих моделей, що дозволяють представити в електронно-обчислювальній машині знання, що зафіксовані у вигляді текстів. Тексти, або сенсова лінгвістична інформація, повинні бути попередньо структуровані або нормалізовані, тобто необхідно в явній формі виділити об'єкти, або поняття, і відносини між ними. При цьому об'єкти (поняття) відповідають вершинам (вузлам) мережі, а відносини між об'єктами характеризуються дугами, що зв'язують вершини. З допомогою семантичної мережі можна, наприклад, описати взаємне розташування об'єктів в кімнаті, якщо використовувати їх назви і задавати відносини між ними зі списку: «поруч», «попереду», «вище», «нижче» тощо [9].

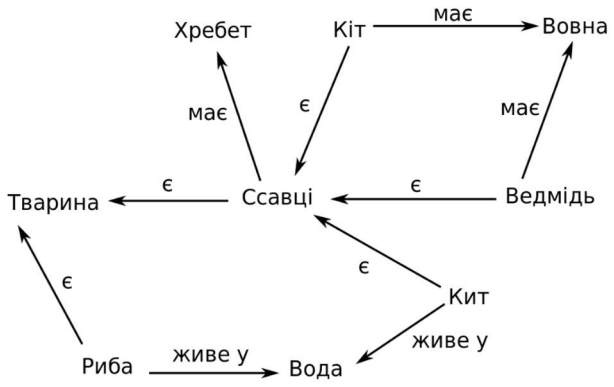


Рисунок 1.3 – Приклад семантичної мережі

Фактично, нейронна та семантична мережі являють собою дві повні протилежності. Так, наприклад, відображення нейронної мережі мало, що скаже дослідникові про природу об'єктів, які він досліджує, настільки вона стає складна вже при невеликій кількості нейронів. Семантична мережа, навпаки, дає прекрасне сенсове уявлення про предмет дослідження, її безпосередньо можна використовувати в якості результату. Нейронна мережа «рахує» досить швидко (виключенням можуть складати лише мережі з зворотними зв'язками), а семантична мережа, в якій обробка – це знаходження всіх дозволених шляхів, при максимально швидких алгоритмах дає квадрат числа вузлів.

Нейронній мережі можна довіряти розпізнавання образів, розрахунки складних функцій і інші алгебраїчні завдання. Семантична мережа числа і математику «не любить», вона з успіхом може виробляти висновок силогізмів, але її дуже важко змусити скласти два числа.

За останні п’ять років, у зв’язку із ростом обчислювальних можливостей, різко зросла популярність застосування згорткових нейронних мереж (ЗНМ).

ЗНМ є спеціальною архітектурою штучних нейронних мереж, запропонованою Яном Лекуном в 1988 році. Вона використовує деякі особливості зорової кори, в якій були відкриті так звані прості клітини, що реагують на прямі лінії під різними кутами, і складні клітини, реакція яких пов'язана з активацією певного набору простих клітин. Таким чином, ідея згорткових нейронних мереж полягає в чергуванні згорткових шарів і субдискретизучих шарів. Структура мережі - односпрямована (без зворотних зв'язків), принципово багатошарова. Для навчання використовуються стандартні методи, найчастіше метод зворотного поширення помилки, основна ідея якого, полягає в поширенні сигналів помилки від виходів мережі до її входів, в зворотному напрямку, навідміну від прямомого поширення сигналів у звичайному режимі роботи. Функція активації нейронів може бути будь-якою, за вибором дослідника [10].

Назву ця архітектура отримала через наявність операції згортки, суть якої в тому, що кожен фрагмент зображення множиться на матрицю (ядро) згортки поелементно, а результат підсумовується і записується в аналогічну позицію вихідного зображення.

Перевагою використання ЗНМ для класифікації біомедичних зображень є простота у підготовці вхідних даних та відносно висока точність класифікації.

У звичайному персептроні, який представляє собою повнозв'язну нейронну мережу, кожен нейрон пов'язаний з усіма нейронами попереднього шару, причому кожен зв'язок має свій персональний ваговий коефіцієнт. У згортковій нейронній мережі в операції згортки використовується лише обмежена матриця ваг невеликого розміру, яку «рухають» по всьому

оброблюваному шару (на самому початку – безпосередньо по вхідному зображенню), тим самим, формуючи після кожного зсуву сигнал активації для нейрона наступного шару з аналогічною позицією.

Тобто для різних нейронів вихідного шару використовуються одна і та ж матриця ваг, яку також називають ядром згортки. Її інтерпретують як графічне кодування якої-небудь ознаки, наприклад, наявність похилої лінії під певним кутом. Тоді наступний шар, що вийшов в результаті операції згортки такою матрицею ваг, показує наявність даної ознаки в оброблюваному шарі і її координати, формуючи так звану карту ознак. Звичайно що в згортковій нейронної мережі набір ваг не один, а ціла гама, що кодує елементи зображення (наприклад лінії і дуги під різними кутами).

При цьому такі ядра згортки не закладаються дослідником заздалегідь, а формуються самостійно шляхом навчання мережі класичним методом зворотного поширення помилки. Прохід кожним набором ваг формує свій власний примірник карти ознак, роблячи нейронну мережу багатоканальною (багато незалежних карт ознак на одному шарі).

Також слід зазначити, що при переборі шару матрицею ваг її пересувають зазвичай не на повний крок (розмір цієї матриці), а на невелику відстань. Так, наприклад, при розмірності матриці ваг 5×5 її зрушують на один або два нейрона (пікселя) замість п'яти, щоб не «переступити» шукану ознаку.

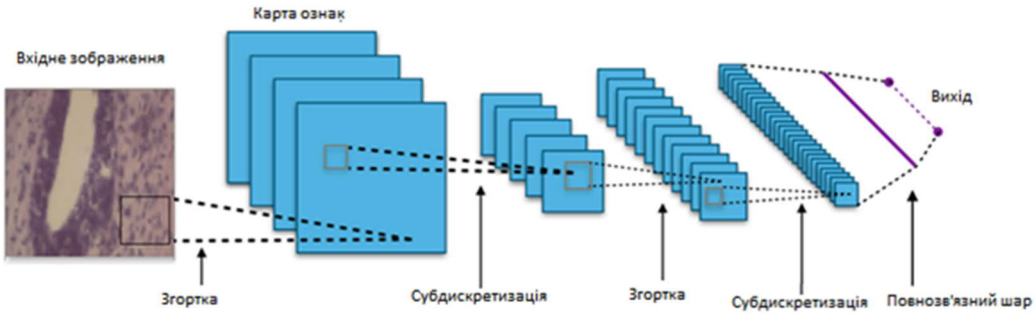


Рисунок 1.4 – Типова структура ЗНМ

Розглянемо типову структуру ЗНМ (рисунок 1.4) більш детально. Мережа складається з великої кількості шарів. Після початкового шару (вхідного

зображення) сигнал проходить серію згорткових шарів, в яких чергується власне згортка і субдискретизація. Чергування шарів дозволяє складати «карти ознак» з карт ознак, на кожному наступному шарі карта зменшується в розмірі, але збільшується кількість каналів. На практиці це означає здатність розпізнавання складних ієрархій ознак. Зазвичай після проходження декількох шарів карта ознак вироджується в вектор або навіть скаляр, але таких карт ознак стає сотні. На виході згорткових шарів мережі додатково встановлюють кілька повнозвязаних шарів (персептрони), на вхід якого подаються кінцеві карти ознак [11].

Послідовність кроків при навчанні ЗНМ зображена на рисунку 1.5.

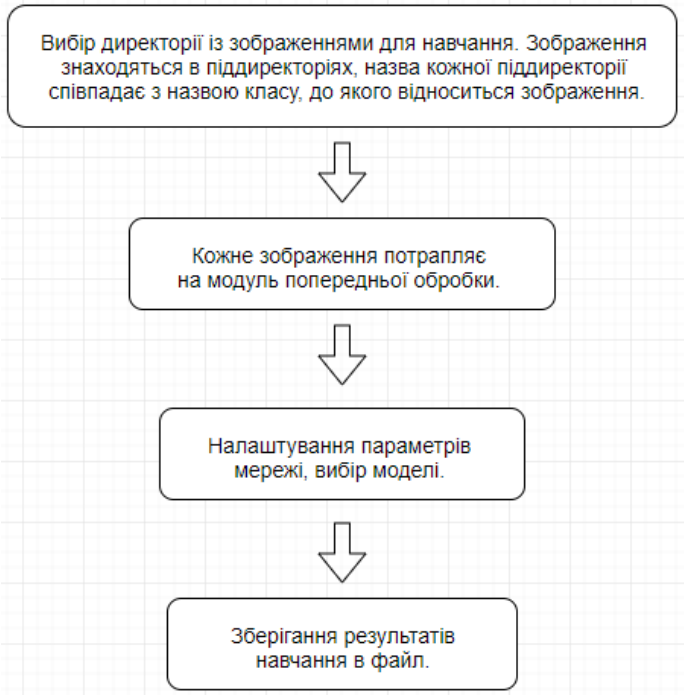


Рисунок 1.5 – Послідовність кроків навчання ЗНМ

Після кількох проходжень згортки зображення і ущільнення за допомогою субдискретизації система перебудовується від конкретної сітки пікселів з високою роздільною здатністю до більш абстрактних карт ознак, як правило на кожному наступному шарі збільшується число каналів і зменшується розмірність зображення в кожному каналі. Зрештою залишається великий набір каналів, що зберігають невелику кількість даних (навіть один

параметр), які інтерпретуються як найабстрактніші поняття, виявлені з вихідного зображення.

Ці дані об'єднуються і передаються на звичайну повнозв'язну нейронну мережу, яка теж може складатися з декількох шарів. При цьому повнозв'язні шари вже втрачають просторову структуру пікселів і мають порівняно невелику розмірність (по відношенню до кількості пікселів вихідного зображення).

Найбільш простим і популярним способом навчання є метод навчання з учителем – метод зворотного поширення помилки і його модифікації.

Послідовність кроків при класифікації зображень з допомогою ЗНМ зображена на рисунку 1.6.

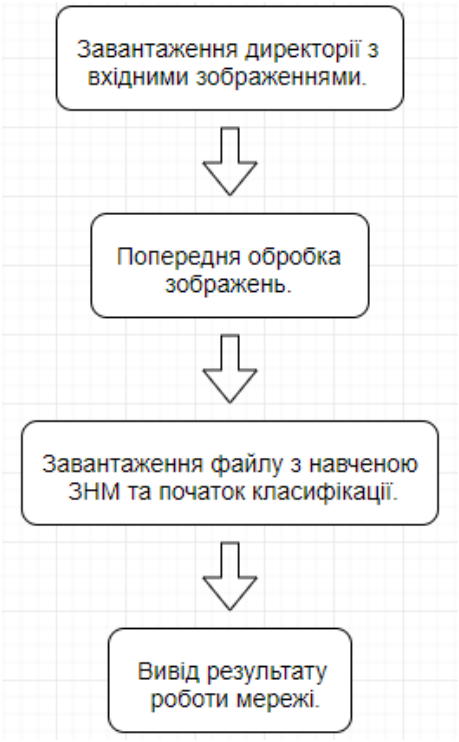


Рисунок 1.6 – Послідовність кроків при класіфікації зображень з допомогою ЗНМ

Метод k-найближчих сусідів (k-nearest neighbors method, k-NN) – один з методів вирішення задачі класифікації. Передбачається, що вже є якась кількість об'єктів з точною класифікацією (тобто для кожного них точно відомо, якого класу він належить) [12]. Потрібно виробити правило, що дозволяє

віднести новий об'єкт до одного з можливих класів (тобто самі класи відомі заздалегідь).

В основі k-NN лежить таке правило: об'єкт вважається належним того класу, до якого належить більшість його найближчих сусідів. Під «сусідами» тут розуміються об'єкти, близькі до досліджуваного в тому чи іншому сенсі.

Зауважимо, що тут необхідно вміти визначати, наскільки об'єкти близькі один до одного, тобто вміти вимірювати «відстань» між об'єктами. Це не обов'язково евклідова відстань. Це може бути міра близькості об'єктів, наприклад, за кольором, формою, смаком, запахом, інтересам (якщо мова йде про формування груп людей), особливостям поведінки і т.д. Отже, для застосування методу kNN в просторі ознак об'єктів повинна бути введена деяка метрика (тобто функція відстані). Передбачається, що об'єкти з близькими значеннями одних ознак будуть близькі і за іншими ознаками (тобто стосуватися одного і того ж класу).

Логістична регресія – це статистична модель, використовувана для прогнозування ймовірності виникнення деякої події шляхом підгонки даних до логістичної кривої. Логістична регресія є одним з статистичних методів класифікації з використанням лінійного дискримінанту Фішера. Також вона входить в топ часто використовуваних алгоритмів в науці про дані.

На відміну від звичайної регресії, в методі логістичної регресії не проводиться пророкування значення числової змінної виходячи з вибірки вихідних значень. Замість цього, значенням функції є ймовірність того, що дане початкове значення належить до певного класу.

Основна ідея логістичної регресії полягає в тому, що простір вихідних значень може бути розділений лінійною границею на дві області, відповідних класам.

Отже, що ж мається на увазі під лінійною границею? У разі двох вимірів – це просто пряма лінія без вигинів. У разі трьох – площину, і так далі. Ця межа задається в залежності від наявних вихідних даних і навчального алгоритму. Щоб все працювало, точки вихідних даних повинні розділятися лінійною

границею на дві вищезазначених області. Якщо точки вихідних даних задовольняють цій вимозі, то їх можна назвати лінійно розділюваними.

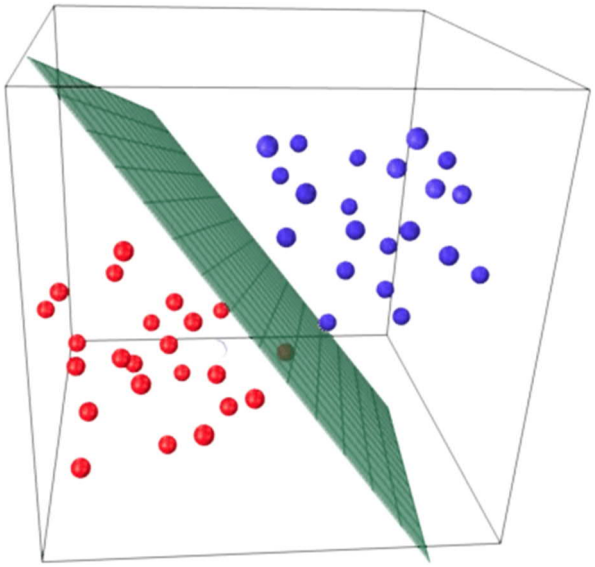


Рисунок 1.7 – Розділююча площина

Зазначена на рисунку 1.7 розділююча площина називається лінійним дискримінантом, оскільки вона є лінійною з точки зору своєї функції, і дозволяє моделі проводити розподіл, дискримінацію точок на різні класи [13].

У таблиці 1.2 наведені переваги та недоліки методів класифікації зображень.

Таблиця 1.2 – Переваги та недоліки методів класифікації зображень

Метод	Переваги	Недоліки
Нейронна мережа	ефективний для великого набору даних; справляється з шумами; метод заснований на даних; адаптування до змін навколишнього середовища; надвисока швидкодія.	високі обчислювальні витрати; попередня підготовка займає багато часу; вважається семантично бідним методом; проблема перенавчання.

Продовження таблиці 1.2

Логістична регресія	містить багато способів для впорядкування моделі; не потрібно перейматися тим, як взаємодіють функції; хороша ймовірнісна інтерпретація; наявність ефективного інструменту оцінки якості моделей - ROC-аналізу	проблема перенавчання; оскільки метод вимагає, щоб кожна точка даних була незалежною від всіх інших точок даних, то якщо спостереження пов'язані один з одним, модель буде схильна переоцінювати значимість цих спостережень.
Згорткова нейронна мережа	менша кількість настроюються вагових коефіцієнтів; узагальнення демонстрованої інформації; часткова інваріантність до масштабу за рахунок субдискретизації;	тривалий час навчання з числом шарів згортки понад двох; необхідність у великій кількості прикладів для навчання; занадто багато змінних параметрів мережі.
k-найближчих сусідів	простота реалізації; результат роботи легко піддається інтерпретації; можливість значної модифікації алгоритму.	неефективна витрата пам'яті; пошук найближчого сусіда передбачає порівняння об'єкту, що класифікується з усіма об'єктами вибірки, що вимагає лінійної по довжині вибірки числа операцій.

1.3 Аналіз програмного забезпечення для класифікації

Класифікація даних в широкому сенсі визначається як процес організації даних за відповідними категоріями, щоб їх можна було використовувати і захищати ефективніше. На базовому рівні процес класифікації спрощує пошук і вилучення даних. Класифікація даних має особливе значення, коли мова йде про управління ризиками, дотримання вимог і безпеки даних.

Класифікація даних включає в себе тегування даних, щоб зробити їх легко доступними для пошуку і відстеження, а також усуває множинне дублювання даних, що може знизити витрати на зберігання і резервне копіювання, прискорюючи процес пошуку.

В даному підрозділі буде розлянено декілька найпоширеніших інструментів для класифікації даних.

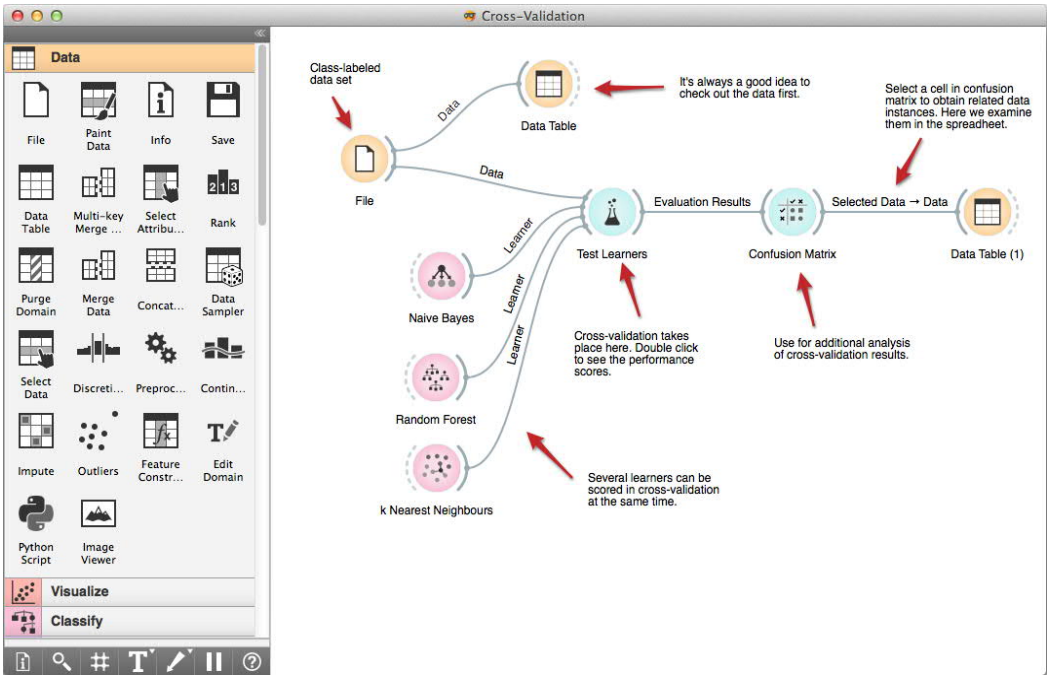


Рисунок 1.8 – Інтерфейс Orange

Інструмент інтелектуального аналізу даних Orange (див. рисунок 1.8) існує вже більше 20 років і є проектом університету Любляни. Ядро

програмного забезпечення було написано на C ++, але з часом програма була розширена мовою програмування Python, який тепер використовується в якості мови запитів. Більш складні операції все ще виконуються в C ++. Orange пропонує корисні додатки для аналізу даних і тексту, а також функції для машинного навчання. Коли справа стосується інтелектуального аналізу даних, він працює з операторами для класифікації, регресії, кластеризації і багато чого іншого.

В Orange, процес аналізу даних може бути розроблений за допомогою візуального програмування. Orange запам'ятовує вибір і пропонує найчастіше використовувані комбінації. Orange має особливості для різних візуалізацій, таких як діаграми розсіювання, стовпчикові діаграми, дерева, дендрограми, мережі і теплові карти. Поєднуючи різні віджети, можна зробити створити структуру аналітики даних. Існує понад 100 віджетів з охопленням більшості стандартних завдань аналізу даних та спеціалізованих додатків.

Написаний на мові програмування Java, RapidMiner пропонує розширену аналітику через засновані на шаблонах фреймворки. Запропонований як сервіс, а не як частина локального програмного забезпечення, цей інструмент займає верхню позицію в списку інструментів інтелектуального аналізу даних.

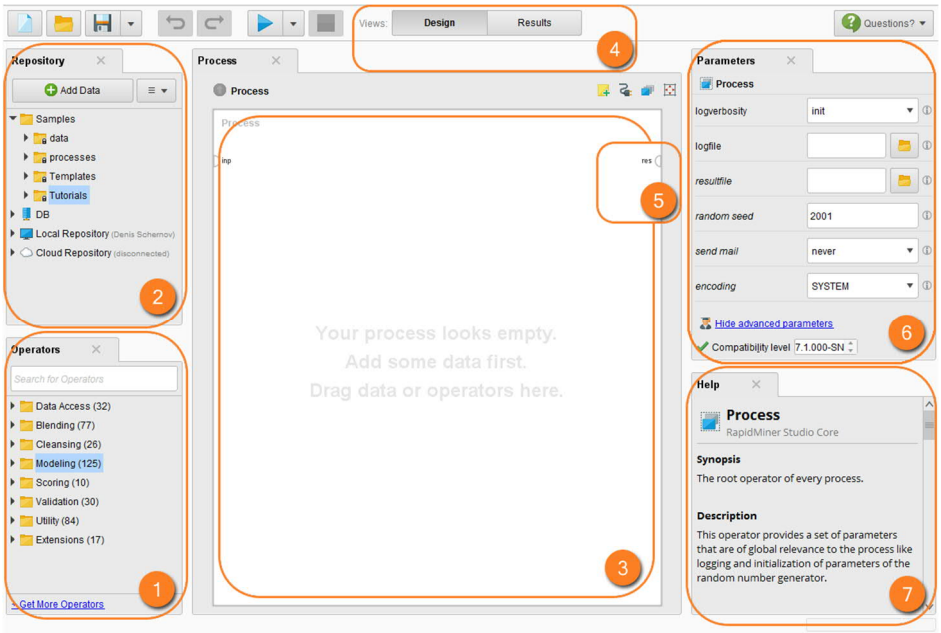


Рисунок 1.9 – Головний екран RapidMiner

RapidMiner – одна з кращих систем прогнозного аналізу, розроблена компанією з такою ж назвою RapidMiner, написана на мові програмування Java і забезпечує інтегроване середовище для глибокого навчання, інтелектуального аналізу тексту, машинного навчання та прогнозного аналізу. Цей інструмент може використовуватися для широкого спектру додатків, в тому числі для бізнес-додатків, комерційних додатків, навчання, освіти, досліджень, розробки додатків, машинного навчання [14].

Weka – це безкоштовне програмне забезпечення з відкритим вихідним кодом на основі Java і доступне для використання в Linux, Mac OS X і Windows.

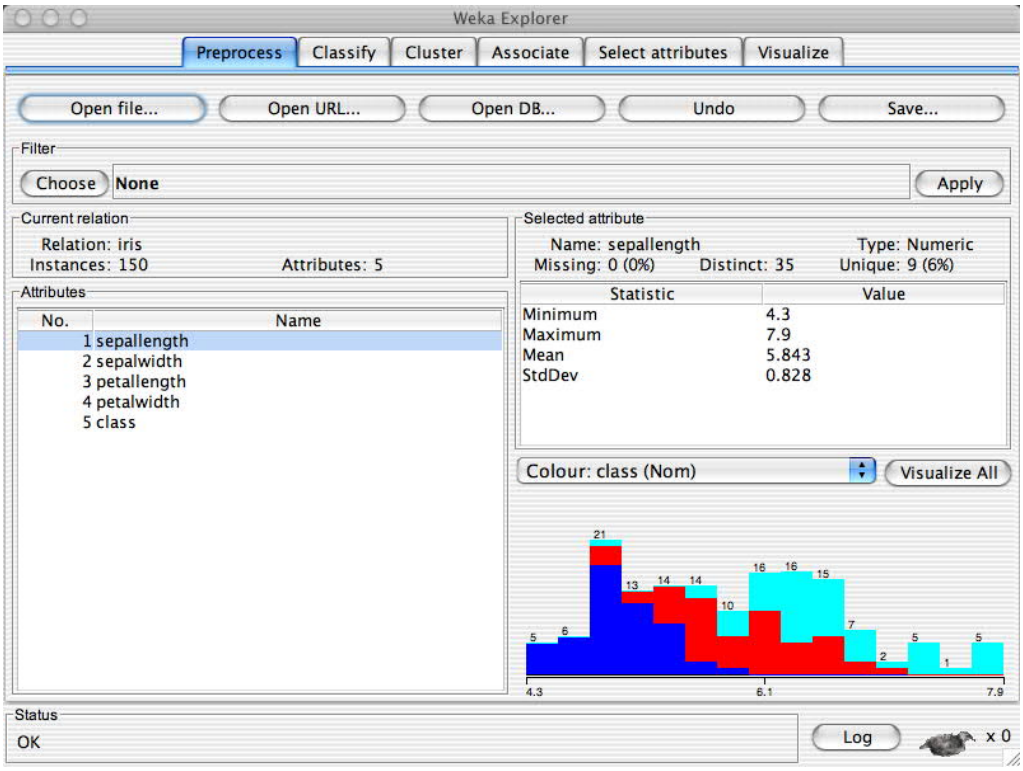


Рисунок 1.10 – Інтерфейс Weka

Weka був створений дослідниками університету Вайкато (Нова Зеландія). Він включає в себе набір алгоритмів машинного навчання для інтелектуального аналізу даних, а також містить інструменти для регресії, кластеризації, асоціації, попередньої обробки даних, класифікації та візуалізації [15].

В таблиці 1.3 наведені переваги та недоліки методів класифікації зображень.

Таблиця 1.3 – Переваги та недоліки програмного забезпечення для класифікації

ПЗ	Переваги	Недоліки
Weka	незалежність від платформи; безкоштовне і з відкритим вихідним кодом; простота використання; -великий збірник методів моделювання і класифікації.	можуть бути проблеми з обробкою відносно великих об’ємів даних; неповна документація; деякі можливості доступні лише з командної стрічки.
Orange	приваблива візуалізація даних; численні онлайн-підручники; безкоштовне і з відкритим вихідним кодом; місить додатки для розширення функціоналу; кросплатформенний графічний інтерфейс; має хороший відладчик.	займає досить багато пам’яті, оскільки потрібно ще встановити QT; обмежений список алгоритмів машинного навчання; слабке в класичній статистиці; можливості звітів обмежені.
RapidMiner	простота установки; сильна візуалізуалізація; кілька варіантів розгортання в залежності від уподобань; точна попередня обробка даних;	важке для нових користувачів; займає багато пам’яті і тим самим сповільнює роботу системи; відносно менше форумів для підтримки.

1.4 Висновки та постановка задачі

В даному розділі проведено аналітичний огляд класифікації біомедичних зображень, засобів класифікації даних та програмного забезпечення для класифікації, що дозволило обрати найоптимальніший шлях для написання нейронної мережі.

Метою даного дипломного проекту є розробка нейронної мережі для класифікації біомедичних зображень.

Для досягнення поставленої мети потрібно виконати наступні задачі:

– провести більш детальше дослідження нейромережевого методу класифікації даних;

– отримати теоретичні знання з класифікації бінарних зображень;

– ознайомитися з одношаровим та багатошаровим персептронами;

– ознайомитись з алгоритмами класифікації гістологічних та цитологічних зображень;

– використати тестову та навчальну вибірки;

– розробити структуру бази даних нейромережі;

– навчити нейромережу класифікувати біомедичні зображення;

– порівняти результати роботи нейронної мережі з аналогами.

В даному розділі було досліджено основні та найбільш популярні засоби для класифікації даних. Зокрема, проведено огляд сучасних і найпопулярніших методів класифікації даних. Результат проведеного дослідження показав частину інформації про кожен метод класифікації, його переваги і недоліки. Також було розглянено класифікацію біомедичних зображень та досліджено програмне забезпечення для класифікації динх, проведено порівняльний аналіз відомого програмного забезпечення, проаналізовано основні переваги та недоліки досліджуваного програмного забезпечення.

2 АЛГОРИТМИ НЕЙРОМЕРЕЖЕВОЇ КЛАСИФІКАЦІЇ БІОМЕДИЧНИХ ЗОБРАЖЕНЬ

2.1 Алгоритми роботи одношарового та багатошарового персептронів

Нервова система людини складається з клітин, які називаються нейронами, і має надзвичайну складність: близько 10^{11} нейронів беруть участь у близько 10^{15} передавальних зв'язках, що мають довжину метр і більше. Кожен нейрон має багато якостей, спільних з іншими клітинами, але його унікальною здатністю є прийом, обробка і передача електрохімічних сигналів по нервовим шляхам, що утворюють комунікаційну систему мозку.

Структура нейрона має вигляд, представлений на рисунку 2.2. Відповідно, маємо формулу 2.1 для знаходження зваженої суми вхідних сигналів нейрона, а також передавання цієї суми, як аргументу, в функцію активації згідно формули 2.2.

$$S = \sum_{i=1}^{n} w_i x_i - T, \tag{2.1}$$

де T – поріг нейрона (у багатьох моделях обходяться без нього);

$x_1, x_2, ..., x_n$ – значення, що надходять на входи (синапси) нейрона;

$w_1, w_2, ..., w_n$ – ваги синапсів, які можуть бути як гальмуючими, так і підсилюючими;

S – зважена сума вхідних сигналів.

$$y = F(S) \tag{2.2}$$

де F – функція активації нейрона.

Одношаровий персептрон є одним з найпростіших варіантів нейронної мережі (рисунок 2.2) і містить лише один нейрон. Будучи самостійною моделлю нейронної мережі з одного боку, одношаровий персептрон є основним конструкційним елементом для більшості моделей нейронних мереж, з іншого боку [16].

На вхід одношарового персептрона надходить набір вхідних сигналів $x_1, x_2, ..., x_n$. Кожен вхідний сигнал помножується на відповідну вагу зв'язку $w_1, w_2, ..., w_n$ – аналог ефективності синапсу (міжнейронного контакту).

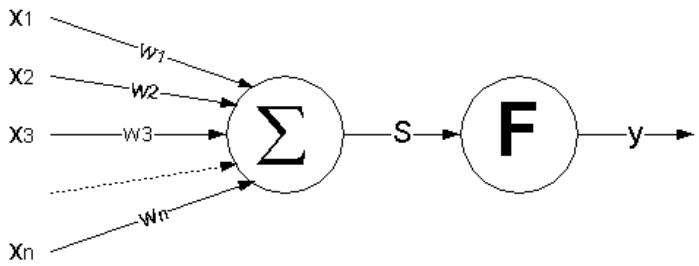


Рисунок 2.1 – Одношаровий персептрон

Вага зв'язку є скалярною величиною, позитивною для збудливих і негативною для гальмуючих зв'язків. Зважені вагами зв'язків вхідні сигнали надходять на блок сумації, що відповідає тілу клітини, де здійснюється їхня алгебраїчна сумація та визначається рівень збудження нейроподібного елемента.

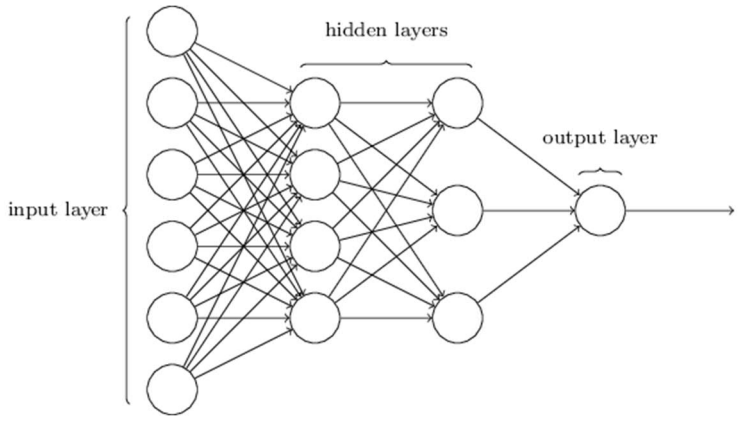


Рисунок 2.2 – Нейронна мережа типу багатошаровий персептрон

Багатошаровими персептронами (див. рисунок 2.2) називають нейронні мережі прямого поширення. Вхідний сигнал в таких мережах поширюється в прямому напрямку, від шару до шару. Багатошаровий персептрон в загальному уявленні складається з наступних елементів:

– безлічі вхідних вузлів, які утворюють вхідний шар;

– одного або декількох прихованих шарів обчислювальних нейронів;

– одного вихідного шару нейронів.

Відповідно, вхідний шар отримує інформацію, n прихованих шарів (зазвичай їх не більше 3) інформацію обробляють, а вихідний шар виводить результат. У кожного з нейронів є 2 основні параметри: вхідні дані (input data) і вихідні дані (output data) [17]. У разі вхідного нейрона: input = output. В інших, в поле input потрапляє сумарна інформація всіх нейронів з попереднього шару, після чого, вона нормалізується, за допомогою функції активації і потрапляє в поле output. Важливо пам’ятати, що нейрони оперують числами в діапазоні [0,1] або [-1,1].

Синапс, як вже було вияснено, це зв’язок між двома нейронами. У синапсів є 1 параметр – вага. Завдяки йому, вхідна інформація змінюється, коли передається від одного нейрона до іншого.

Припустимо, є 3 нейрона, які передають інформацію далі. Тоді у нас є 3 ваги, відповідні кожному з цих нейронів. У того нейрона, у якого вага буде більше, та інформація і буде домінуючою в наступному нейроні.

Насправді, сукупність ваг нейронної мережі або матриця ваг – це своєрідний мозок всієї системи. Саме завдяки цим вагам, вхідна інформація обробляється і перетворюється в результат. Важливо пам’ятати, що під час ініціалізації нейронної мережі, ваги розставляються в випадковому порядку [18].

Функція активації є спосібом нормалізації вхідних даних. Тобто, якщо на вході у вас буде велике число, то пропустивши його через функцію активації, ви отримаєте вихід в потрібному вам діапазоні. Головні відмінності функцій активації – діапазон значень.

Вид функції активації може мати різне математичний вираз, вибір якого визначається характером вирішуваних завдань. Розглянемо формули декількох найпоширеніших функцій активації: сигмоїдна функція (формула 2.3), лінійна функція (формула 2.4), функція гіперболічного тангенса (формула 2.5), логарифмічна функція (формула 2.6).

$$y = \frac{1}{1 + e^{-CS}}\ , \tag{2.3}$$

де $C > 0$ – коефіцієнт ширини сигмоїди по осі абсцис;

S – зважена сума вхідних сигналів.

$$y = k \times S\,, \tag{2.4}$$

де k – кутовий коефіцієнт прямої;

S – зважена сума вхідних сигналів.

$$y = th(CS) = \frac{e^{CS} - e^{-CS}}{e^{CS} + e^{-CS}}, \tag{2.5}$$

де $C > 0$ – коефіцієнт ширини сигмоїди по осі абсцис;

S – зважена сума вхідних сигналів.

$$y = \ln(S + \sqrt{S^2 + 1})\,, \tag{2.6}$$

де S – зважена сума вхідних сигналів.

Графік деяких функцій активації представлений на рисунку 2.3.

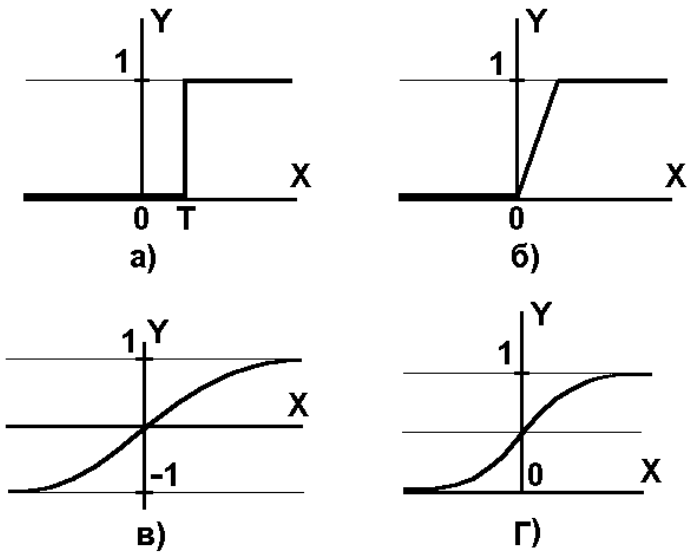


Рисунок 2.3 – Графіки функцій активації: а) функція одиничного стрибка; б) лінійний поріг; в) гіперболічний тангенс; г) сигмоїд

Лінійна функція активації вже майже ніколи не використовується, за винятком випадків, коли потрібно протестувати нейронну мережу або передати значення без перетворень.

Логарифмічна функція активації характерна тим, що має діапазон [-∞; +∞] з точкою перегину на початку координат, підсилює дуже слабкі сигнали і послаблює дуже сильні.

Сигмоїда є найпоширенішою функцією активації і саме на ній показано більшість прикладів в мережі. Інша її назва – логістична функція. Сигмоїда добре показує себе в задачах класифікації. Вона приймає на вході довільне дійсне число, а на виході дає дійсне число в інтервалі від 0 до 1. Зокрема, великі (по модулю) негативні числа перетворюються в нуль, а великі позитивні – в одиницю. Історично сигмоїда знаходила широке застосування, оскільки її вихід добре інтерпретується, як рівень активації нейрона: від відсутності активації до повністю насиченої активації. На даний момент сигмоїда втратила свою колишню популярність і використовується доволі рідко, оскільки має два серйозні недоліки:

1. Насичення сигмоїди призводить до загасання градієнтів.

2. Вихід сигмоїди не центрований щодо нуля.

Гіперболічний тангенс є одним з прикладів сигмоїдальної функції активації. Ця функція приймає на вході довільне дійсне число, а на виході дає дійсне число в інтервалі від -1 до 1. Подібно сигмоїді, гіперболічний тангенс може насичуватися. Однак, на відміну від сигмоїди, вихід даної функції центрований щодо нуля, а отже на практиці завжди краще використовувати гіперболічний тангенс, а не сигмоїду [19].

На сьогоднішній день існує багато різновидів нейронних мереж, характерних для різних типів завдань, які можна класифікувати за такими ознаками [20]:

– тип вхідної інформації (аналогові або двійкові дані);

– характер навчання (з учителем і без);

– характер настройки синапсів (фіксовані або динамічні зв'язки);

– метод навчання (зворотне поширення, конкурентна настройка, використання правила Хебба, гібридні мережі і т.д.);

– характер зв'язків (пряме і пряме-зворотне поширення інформації);

– архітектура (персептрони, рециркуляційні, рекурентні, зустрічного поширення, з радіально-базисною функцією активації і т.п.).

Для вирішення завдань за допомогою нейромереж необхідно:

1. Розглянути поставлену задачу і подумати, а чи не простіше її вирішити за допомогою інших методів.

2. Визначитися з початковим пулом значущих входів. Пул входів зазвичай перебирається протягом часу і вибираються ті, на яких дається найменша помилка.

3. Вибрати, що буде на виході нейронної мережі, тобто визначити вихідні змінні.

4. Визначитися з початковою кількістю шарів і нейронів у них (знову ж кількість шарів і нейронів у них зазвичай перебирається в перебігу навчання).

5. Згенерувати мережу і навчити її на заданих прикладах.

2.2 Алгоритми класифікації зображень

В даному підрозділі по чергово розглянемо три алгоритми класифікації зображень, які будуть будуть використовуватися в даному проекті, а саме: нейронні мережі (на основі багатошарового персептрону), наївний баєсівський класифікатор та метод опорних векторів.

Метод опорних векторів відноситься до бінарних класифікаторів, хоча існують способи змусити його працювати і для задач мультікласифікації [21].

Ідею методу зручно проілюструвати на наступному простому прикладі: дані точки на площині, розбиті на два класи (рисунок 2.4). Проведемо лінію, що розділяє ці два класи (червона лінія на рисунку 2.4). Далі, всі нові точки (не з навчальної вибірки) автоматично класифікуються наступним чином: точка вище прямої потрапляє в клас A, точка нижче прямої – в клас B [22].

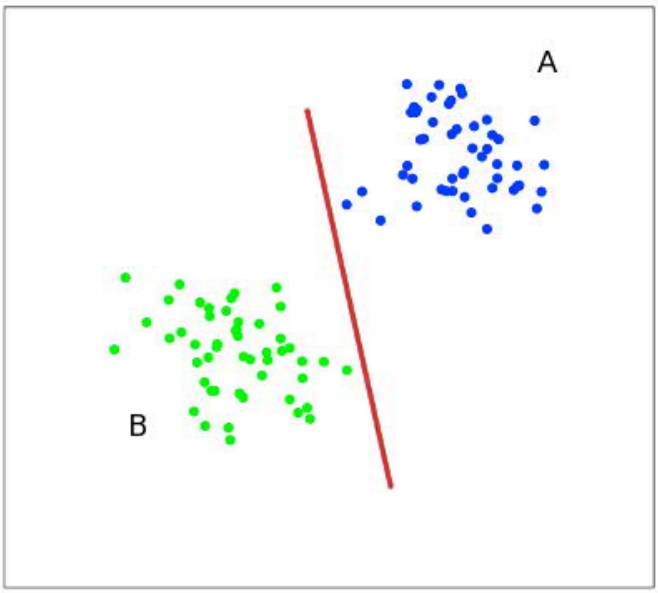


Рисунок 2.4 – Пряма що розділяє два класи

Таку пряму назвемо розділяючою прямою. Однак, в просторах високих розмірностей пряма вже не буде розділяти наші класи, так як поняття «нижче прямої» або «вище прямої» втрачає будь-який сенс. Тому замість прямих

необхідно розглядати гіперплощини – простори, розмірність яких на одиницю менше, ніж розмірність початкового простору.

У нашому прикладі існує кілька прямих, які поділяють два класи (рисунок 2.5). З точки зору точності класифікації найкраще вибрати пряму, відстань від якої до кожного класу максимальна. Іншими словами, виберемо ту пряму, яка розділяє класи найкращим чином (червона пряма на рисунку 2.5). Така пряма, а в загальному випадку – гіперплощина, називається оптимальною розділяючою гіперплощиною.

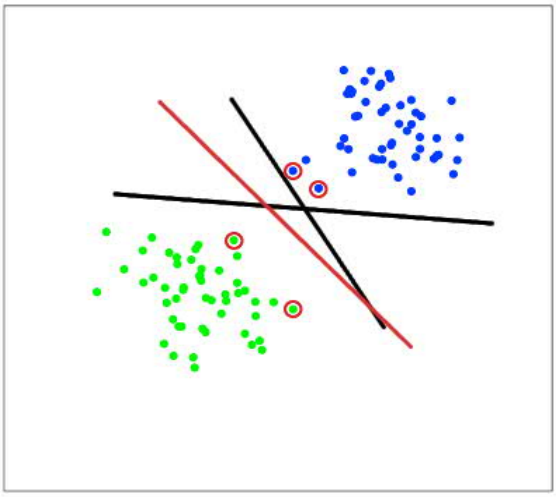


Рисунок 2.5 – Три прямих що розділяють два класи

Вектора, що лежать ближче всіх до розділяючої гіперплощини, називаються опорними векторами. На рисунку 2.5 вони позначені червоним [23].

Наївний байєсівский алгоритм – це алгоритм класифікації, заснований на теоремі Байєса з припущенням про незалежність ознак. Іншими словами, НБА передбачає, що наявність якої-небудь ознаки в класі не пов'язана з наявністю будь-якої іншого ознаки. Наприклад, фрукт може вважатися яблуком, якщо він червоний, круглий і його діаметр становить близько 8 сантиметрів. Навіть якщо ці ознаки залежать одна від однї або від інших ознак, в будь-якому випадку вони вносять незалежний внесок у ймовірність того, що цей фрукт є яблуком. У зв'язку з таким припущенням алгоритм називається «наївним» [24].

Теорема Байєса дозволяє розрахувати апостеріорну ймовірність $P(c|x)$ на основі $p(c)$, $P(x)$ і $P(x|c)$ за формулою 2.7 [25].

$$P(c|x) = \frac{P(c|x)P(c)}{P(x)}, \tag{2.7}$$

де $P(c)$ – апріорна ймовірність даного класу;

$P(c|x)$ – апостеріорна ймовірність даного класу c (тобто даного значення цільової змінної) при даному значенні ознаки x;

$P(x|c)$ – правдоподібність, тобто ймовірність даного значення ознаки при даному класі;

$P(x)$ – апріорна ймовірність даного значення ознаки.

Моделі на основі НБА досить прості і вкрай корисні при роботі з дуже великими наборами даних. При своїй простоті НБА здатний перевершити навіть деякі складні алгоритми класифікації.

Про нейронні мережі та багатошаровий персептрон вже було сказано достатньо. Тепер можна покроково описати алгоритм роботи сайту.

Перший крок алгоритму – завантаження користувачем файлів навчальної вибірки. Щоб запобігти отриманню помилок та дивної поведінки сайту, на сервері було описано валідацію для завантажуваних файлів, а саме: перевірка розширення (дозволені розширення txt та arff) та перевірка розміру (максимальний розмір файлу 5 мегабайт). Завдяки даним покращенням робота сайту стає більш передбачуваною.

Після завантаження браузером вибраних користувачем файлів інтерфейс вибору файлів зникає і появляжться новий – блок з назвам вибраних файлів та випадаюче меню, в якому користувачу потрібно вибрати метод класифікації моделі. Перевіривши правильность вибраних файлів та класифікатора користувач нажимає кнопку тренувати, після чого відбудеться обробка даних на сервері.

Код на сервері виконає валідацію файлів і при наявності хоча б однієї з двох оброблюваних помилок завантажуваних файлів до клієнта буде відправлено відповідне повідомлення, яке буде виведено на екран.

Якщо файли проходять валідацію виконається тренування моделі даних та її запис у файл data.txt, що розташований в корені папки з серверною частиною додатку. Залежно від вибраного класифікатора, буде виконано відповідний код на сервері. Цю логіку виконує конструкція switch, яка містить перевірку по отриманому від клієнта класифікатора та три опції (case), у кожній з яких створюватиметься об’єкт класу обраного класифікатора.

Кожен з класифікаторів звичайно ж, відрізняється способом створення, а точніше аргументами що приймає клас під час створення.

Наївний баєсівський класифікатор, представлений класом NaiveBayes, не приймає жодного агрументу.

Нейронну мережу представляє клас Multilayer Perceptron Classifier (MLPClassifier), конструктор якого приймає три аргумента. Перший аргумент це кількість вхідних шарів, другим – масив з налаштуваннями прихованих шарів, де кожне значення представляє число нейронів у кожному прихованому шарі. До цього значення також може бути додана певна функція активації цього нейрона.

Третім аргументом іде масив з усіма назвами класів. Метод опорних векторів представляє клас SVC. Його конструктор приймає досить таки багато параметрів, основним з яких є тип ядра, від якого буде залежати точність класифкації.

Після конструкції switch в коді вже точно існуватиме змінна, що представляє собою класифікатор. Незалежно від вибраного методу класифкації, у цієї змінної є два найважливіших метода: train та predict, відповідно тренування та прогнозування. Оскільки зараз на сервері відбувається обробка запиту на тренування, то використовується і відовідниий метод. Метод train приймає першим аргументом масив з даними для тренування та масив з назвами класів. Кількість елементів у цих двох масиває повинна співпадати і саме

завдяки цим двом аргументам класифікатор зможе навчитися розрізняти різні класи цитологічних та гістологчних зображень. Результат виконання тренування буде записаний в файл data.txt, щоб при виконанні іншого методу – прогнозування, дані з цього файлу були зручно отримані.

Після виконання тренування, клієнта буде інформовано про успішне тренування або ймовірні валідаційні помилки і при відсутності помилок користувач буде переадресований на сторінку з прогнозуванням. На даній сторінці користувач також виконує завантаження файлів, на цей раз – з тестової вибірки, після чого нажимає кнопку прогнозування. Серверна сторона, отримавши даний запит також виконає код для валідації файлів.

При відсутності помилок, з файлу data.txt, що вже містить дані, тренованої швидше моделі, буде витягнено усі дані, після чого виконається прогнозування. В призначений для цього метод – predict потрібно передати один аргумент – масив, що включає вже натреновані дані.

Для того щоб дізнатися відсоток відгаданих класифікатором класів використовується спеціальний клас для порівняння елементів масиву з відгаданими класами та елементів іншого масиву, що включають правильні назви класів.

Таким чином на відправлений запит на прогнозування буде отримано відповідь у вигляді стрічки, що показуватиме точність правильно відгаданих файлів тестової вибірки, або при наявності, звичайно ж, стрічку з повідомленням про валідаційну помилку.

2.3 Тестова та навчальна вибірки зображень

Однією з ключових вимог, необхідних для створення успішних проектів машинного навчання, є достовірний набір даних. Набори даних мають

вирішальне значення, оскільки вони дозволять нам навчати наш класифікатор по вже класифікованим прикладам. Дані, що використовуються для побудови остаточної моделі, зазвичай надходять з двох наборів даних (вибірок) і, на щастя, набір даних зображень з явними ознаками захворювання є доступним для виконання даного дипломного проекту.

Модель спочатку поміщається в навчальну вибірку, яка представляє собою набір прикладів, використовуваних для підгонки параметрів (наприклад, ваги з'єднань між нейронами в штучних нейронних мережах) моделі. Модель (наприклад, нейронна мережа або наївний баєсівский класифікатор) навчається на даних з навчальної вибірки з використанням контрольованого методу навчання (наприклад градієнтного спуску).

Тестова вибірка даних – це набір даних, який використовується для об'єктивної оцінки остаточної відповідності моделі даних навчальної вибірки. Тестова вибірка даних являє собою золотий стандарт, який використовується тільки після того, як модель повністю навчена з використанням навчальної вибірки.

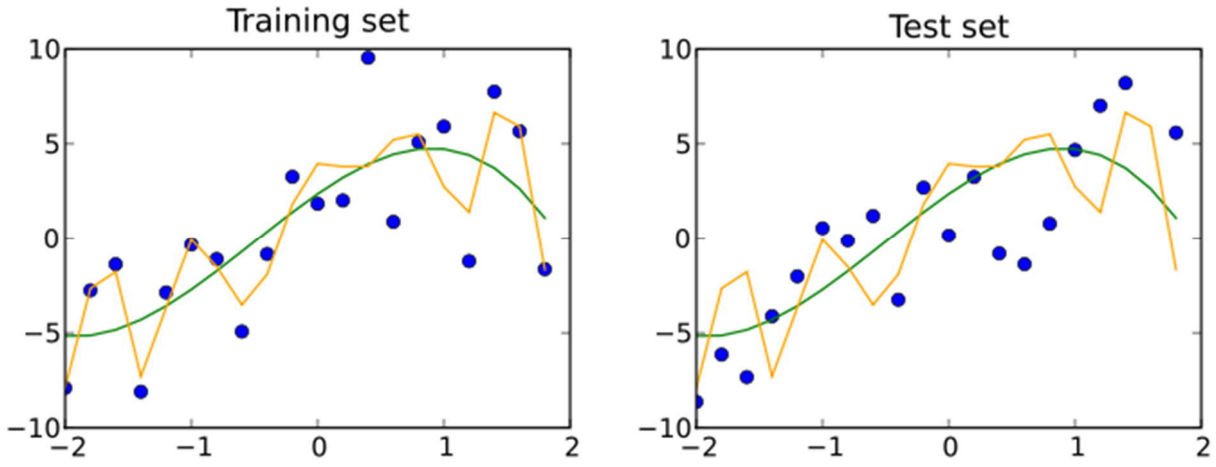


Рисунок 2.6 – Перенасичення моделі

Тестова вибірка даних не залежить від навчальної вибірки даних, але відповідає тому ж розподілу ймовірностей, що і остання. Якщо модель, що відповідає навчальній вибірці, також добре відповідає тестовій вибірці, то це

означає що відбулося мінімальне перенасичення (див. рисунок 2.6). Краща підгонка навчального набору даних в порівнянні з тестовим набором даних зазвичай вказує на перенасичення [26].

На рисунку 2.6 зображено навчальну вибірку (ліворуч) та тестову вибірку (праворуч) з однієї і тієї ж статистичної сукупності у вигляді синіх точок. Дві прогностичні моделі відповідають даним навчання. Обидві підігнані моделі представлені як з навчальними, так і з тестовими вибірками.

Розглянемо набір даних над яким ми будемо працювати. Він містить 4 класи, кожен з яких представляє один тип захворювання і представлений у текстовому форматі arff. Ми працюватимемо з класифікацією таких класів захворювань:

– цитологічне зображення фіброзно-кістозної мастопатії;

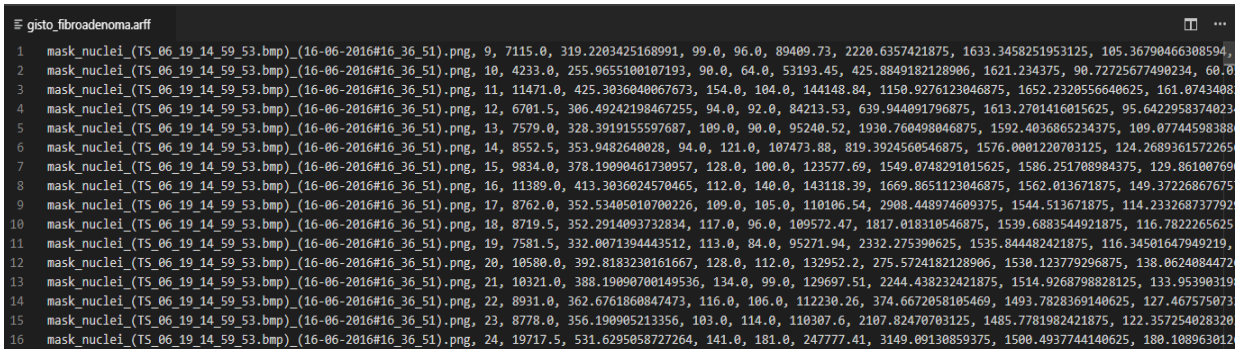
– гістологічне зображення фіброаденоми;

– гістологічне зображення непроліферативної мастопатії;

– гістологічне зображення проліферативної мастопатії.

Усі ці 4 файла містять деяку кількість записів, кожен з яких представляє окреме зображення. Необроблений запис представляє наступні 20 параметрів: img_name, contour_area, contour_perimetr, contour_height, contour_width, contour_circularity, xc, yc, major_axis, minor_axis, theta, equiDiameter, Bx, By, B_width, B_height, B_area, aspect_ratio, roudness, compactness.

На рисунку 2.7 зображено набір даних гістологічних зображень фіброаденоми.


gisto_fibroadenoma.arff

```
1  mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 9, 7115.0, 319.2203425168991, 99.0, 96.0, 89409.73, 2220.6357421875, 1633.3458251953125, 105.36790466308594,
2  mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 10, 4233.0, 255.9655100107193, 90.0, 64.0, 53193.45, 425.8849182128906, 1621.234375, 90.72725677490234, 60.0
3  mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 11, 11471.0, 425.3036040067673, 154.0, 104.0, 144148.84, 1150.9276123046875, 1652.2320556640625, 161.0743408
4  mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 12, 6701.5, 306.49242198467255, 94.0, 92.0, 84213.53, 639.944091796875, 1613.2701416015625, 95.64229583740234
5  mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 13, 7579.0, 328.3919155597687, 109.0, 90.0, 95240.52, 1930.760498046875, 1592.4036865234375, 109.07744598388
6  mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 14, 8552.5, 353.9482640028, 94.0, 121.0, 107473.88, 819.3924560546875, 1576.0001220703125, 124.26893615722656
7  mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 15, 9834.0, 378.19090461730957, 128.0, 100.0, 123577.69, 1549.0748291015625, 1586.251708984375, 129.86100769
8  mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 16, 11389.0, 413.3036024570465, 112.0, 140.0, 143118.39, 1669.8651123046875, 1562.013671875, 149.37226867675
9  mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 17, 8762.0, 352.53405010700226, 109.0, 105.0, 110106.54, 2908.448974609375, 1544.513671875, 114.23326873779
10 mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 18, 8719.5, 352.2914093732834, 117.0, 96.0, 109572.47, 1817.018310546875, 1539.6883544921875, 116.7822265625
11 mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 19, 7581.5, 332.0071394443512, 113.0, 84.0, 95271.94, 2332.275390625, 1535.844482421875, 116.34501647949219,
12 mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 20, 10580.0, 392.8183230161667, 128.0, 112.0, 132952.2, 275.5724182128906, 1530.123779296875, 138.0624084472
13 mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 21, 10321.0, 388.19090700149536, 134.0, 99.0, 129697.51, 2244.438232421875, 1514.9268798828125, 133.95390319
14 mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 22, 8931.0, 362.6761860847473, 116.0, 106.0, 112230.26, 374.6672058105469, 1493.7828369140625, 127.46757507
15 mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 23, 8778.0, 356.190905213356, 103.0, 114.0, 110307.6, 2107.82470703125, 1485.7781982421875, 122.35725402832
16 mask_nuclei_(TS_06_19_14_59_53.bmp)_(16-06-2016#16_36_51).png, 24, 19717.5, 531.6295058727264, 141.0, 181.0, 247777.41, 3149.09130859375, 1500.4937744140625, 180.10896301
```

Рисунок 2.7 – Набір даних гістологічних зображень фіброаденоми.

Кожен з 4 файлів містить різну кількість записів з даними – від 5000 до 11000 тисяч. Але щоб не було перенасичення одним класом потрібно взяти їх однакову кількість з кожного файлу. Ми візьмемо по 3700 записів для навчальної і по 925 записів для тестової вибірок.

Відношення кількості даних навчальної і тестової вибірок залежить від доволі багатьох факторів, але загальним рішенням є таке, що якщо набір даних є відносно великим, то варто виділяти 75% набору даних на навчальну вибірку і 25% на тестову вибірку, а якщо набір даних не є великим, то принято виділяти 90% на навчальну вибірку і 10% на тестову. Відношення кількості даних навчальної і тестової вибірок в даному проекті складає 75% до 25%, оскільки я вважаю що набір даних у нас великий, більше 18 тисяч записів, що, відповідно складає 18 тисяч зображень.

Тепер, маючи таку кількість доступних нам записів, у нас є більше даних, ніж нам потрібно для нашого прикладу; для практичних цілей нас цікавлять тільки такі параметри записів:

– contour_area;

– contour_perimeter;

– compactness.

Де contour_area – площа контуру зображення, contour_perimeter – периметр контуру зображення, compactness – компактність зображення. Інші параметри можна відкинути, оскільки вони будуть спотворювати класифікацію через невелику відмінність в значеннях.

3. ПРОГРАМНИЙ МОДУЛЬ СИСТЕМИ КЛАСИФІКАЦІЇ БІОМЕДИЧНИХ ЗОБРАЖЕНЬ

3.1 Структура програмного модулю

Практичним завданням даного дипломного проекту є створення клієнт-серверного веб-додатку, призначеного для класифікації біомедичних зображень. Створений додадок включає в себе фронтенд та бекенд частини. Фронтенд частина відповідає за публічну частину додатку, з якою безпосередньо контактує і взаємодіє користувач, в той час як бекенд частина відповідає за обробку даних на сервері, і оскільки даних для роботи дуже багато, було б недоречно виконувати цю обробку на клієнтській частині через можливі зависання користувацького інтерфейсу.

Структура фронтенду додатку зображена на рисунку 3.1.

Рисунок 3.1 – Структура файлів на фронтенд частині проекту

Коротко розглянемо використовувані технології в фронтенд частині додатку. HTML є спеціальною мовою розмітки, яка застосовується при створенні сайтів в інтернеті. Браузери прекрасно розуміють HTML і можуть інтерпретувати його в зрозумілому вигляді. CSS – це набір параметрів (стилів) форматування, який застосовується до елементів HTML документа, щоб змінити їх зовнішній вигляд. JavaScript – це мова програмування, що дозволяє створювати скрипти, які вбудовуються в HTML-сторінки і виконуються в браузері відвідувача сторінки. Для простішої та ефективнішої розробки буде використовуватися Javascript бібліотека React.js, що й представляє структуру фронтенду (див. рисунок 3.1). Також буде використана бібліотека Axios для відправки HTTP запитів на створений нами сервер.

Перейдемо до опису файлів. Одним з найважливіших файлів будь-якому сучасному фронтенд проекті є файл package.json. Уявімо ситуацію, коли під час розробки проекту ми встановили кілька npm пакетів. Npm пакетом називається один або кілька JavaScript-файлів, що представляють собою якусь бібліотеку або інструмент. У нашому випадку це бібліотеки React та Axios. Ці пакети лежать у нас в проекті в папці node_modules. У нас на машині все працює, але рано чи пізно нам необхідно запустити проект на іншій машині, або дати іншому розробнику. Далі ми закидаємо наш проект в систему контролю версій, але папку node_modules важливо не чіпати, проігнорувавши її через файл .gitignore, тому що вона велика і займає багато місця. Так виходть що, на іншій машині після скачування того ж проекту з git неможливо запустити, оскільки невідомо які npm пакети ми використовували. Для цього нам і потрібен файл package.json. Він зберігає список пакетів, необхідних для проекту з потрібними версіями, і на іншій машині ми можемо легко встановити всі пакети, які вказані там за допомогою команди npm install.

Файл axios-php.js містить інстанс бібліотеки Axios. Інстанси використовуються для розділення функціоналу цієї бібліотеки на менші частини. В даному проекті цей інстанс буде відповідати лише за HTTP запити пов’язані з основним завданням проекту – відправкою файлів для класифікації

та отримання даних про точність та правильність класифікації на основі вже інших файлів (тестова вибірка). В перспективі може бути доданий інший інстанс, що відповідатиме, наприклад, за авторизацію та автентифікацію користувачів [27].

Далі пройдемось по папкам. Папка config містить конфігураційні файли використовуваних в проекті модулів та npm пакетів (їх у нас, насправді, далеко не два), а також налаштування інструменту webpack та webpack-dev-server. Просто кажучи webpack відповідає за трансформацію усього нашого фронтенд проекту в декілька статичих файлів (HTML, CSS, Javascript, зображення тощо), що будуть відправлятися сервером клієнту. webpack-dev-server робить, по суті, те же саме, але використовується під час розробки для зручності розробникам.

Папка node_modules містить усі npm пакети проекту, тобто залежності проекту, залежності залежностей і т.д.

Папка public містить статичні файли, які рідко будуть змінені під час розробки.

У папці scripts містяться автоматично згенеровані сценарії для запуску, тестування та складання нашого фронтенд додатку.

Найцікавішою зі всіх папок є папка src, в якій відбувається весь процес розробки додатку. Зазвичай всередині папки src розміщується структура React додатку, наприклад папки з файлами компонентів або папки з мультимедійними файлами (зображення, відео тощо). У даному проекті в папці src містяться компоненти для керування логікою додатку, так звані компоненти-контейнери, а також презентаційні компоненти, які немають ніякої логіки і просто рендерять свій вміст. Компонентами-контейнерами в проекті є Trainer.js та Predicter.js. Вони доволі похожі, але відрізнаються призначенням. Перший керує станом додатку при завантаженні файлів для відправки даних на сервер, що слугуватимуть навчальною вибіркою для штучного інтелекту, в той час як інший компонент-контейнер робить щось похоже, але з файлами тестової вибірки. Презентаційні компоненти будуть розглянені в наступному підрозділі.

Структура бекенду (рисунок 3.2), представлена з допомогою серверної мови створення сценаріїв PHP, що являє собою програмне забезпечення з відкритим вихідним кодом. Оскільки PHP, звичайно ж, не включає в себе інструмент для будь-яких проявів штучного інтелекту, для даної мети буде використана бібліотека PHP-ML. З останні два роки PHP-ML стала однією з найпопулярніших і, як кажуть самі творці, пакет є «свіжим підходом до машинного навчання на PHP».

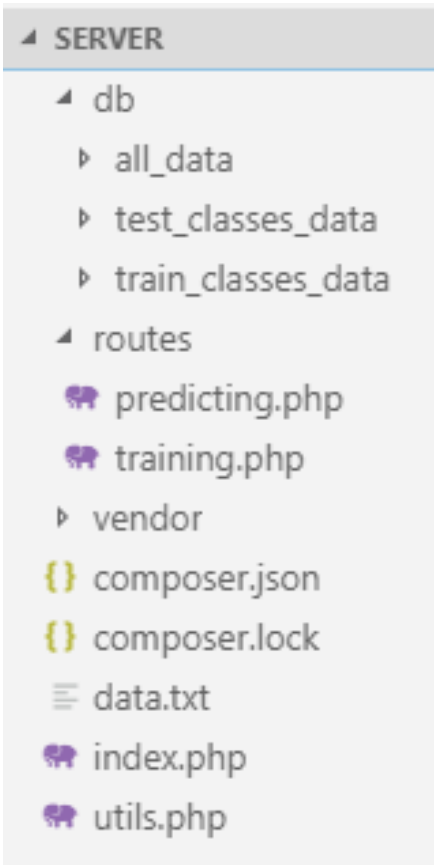


Рисунок 3.2 – Структура серверної частини додатку

Як і на фронтенді, на бекенді також є свій менеджер залежностей який називається Composer. Усі залежності проекту розташовані в файлі composer.json, а залежності залежностей – в composer.lock. Для того щоб встановити усі вказані залежності з composer.json потрібно в консолі виконати команду php composer.phar install. Під час першого такого встановлення створиться папка vendor, яка міститиме усі встановлені залежності локально.

Серед залежностей у проекті лише швидше згадана бібліотека для штучного інтелекту PHP-ML.

Отже з допомогою Composer маємо зручний спосіб встановлення та маніпулювання залежностями і не буде проблемою додати якийсь новий інструмент для серверної сторони.

На верхньому рівні маємо файл index.php, який представляє точку обробки HTTP запитів, що будуть надходити від фронтенду. У файлі перевіряється тип запиту, тобто чи це запит на тренування даних чи на прогнозування і згідно цієї перевірки буде динамічно завантажуватися відповідний файл з папки routes.

Поки що бекенд частина не є обширною або багатою на різні особливості як наприклад авторизація чи робота з БД і тому дана структура має право на життя, але з доданням нового функціоналу вона може значно змінитися.

Замість бази даних, де б зберігалися дані з тренування кожного користувача в даному проекті виступає файл data.txt. При отриманні запиту від клієнта на тренування моделі дані записуються в цей файл. При прогнозуванні дані зчитуються зчитуються з нього.

Файл utils.php містить, поки що, невелику кількість різних допоміжних функцій, які варто варто виділяти в окремі файли як і було зроблено, щоб їх можна було ефективно перевикористовувати в інших файлах.

Папка db містить файли навчальної та тестової вибірок, які для зручності тестування розробником розташовані в папці проекту. Після успішного закінчення розробки проекту дана папка буде видалена, а ці ж дані навчальної та тестової вибірок будуть відправлятися по HTTP від клієнтів, що зайдуть на даний веб-сайт.

Блок-схема роботи розробленого додатку зображена на кресленні БР.КСМ.07121/15.00.000 А2.

Діаграма послідовності розробленого додатку зображена на кресленні БР.КСМ.07121/15.00.000 С2.

3.2 Тестування роботи програмного модулю

В даному розділі буде продемонстрована робота виконаного проекту. Для початку потрібно відкрити програму XAMPP і з її допомогою запустити веб-сервер (рисунок 3.3), котрий прийматиме HTTP запити з фронтенд частини додатку та повертатиме дані, що оброблятимуться з допомогою PHP, наприклад точність прогнозування.

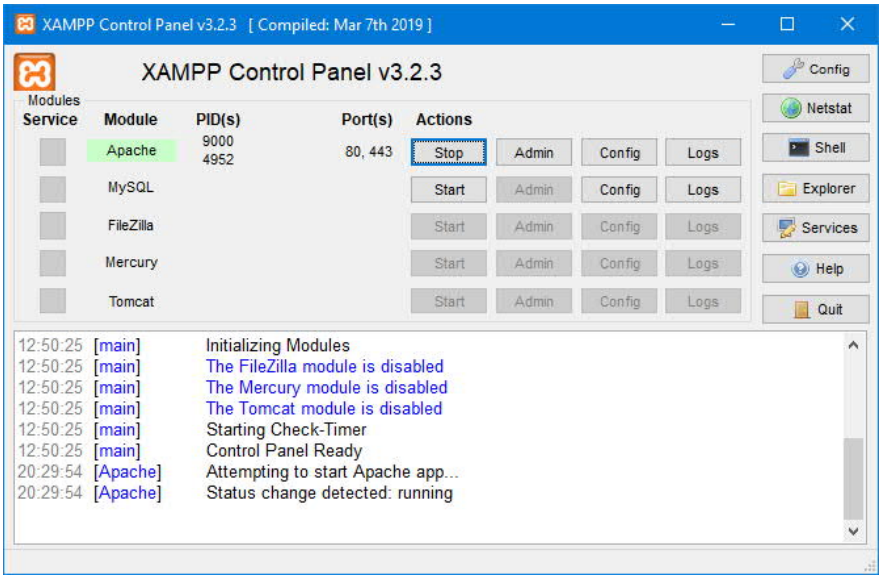


Рисунок 3.3 – Панель управління XAMPP

Далі запускаємо фронтенд частину програми, що керується з допомогою React.js. Ввівши в консолі команду npm start (рисунок 3.4) на порті 3000 буде запущено локальний сервер.

Рисунок 3.4 – Запуск фронтенду

Тепер при відвідуванні сторінки http://www.localhost:3000 можна побачити інтерфейс даного дипломного проекту (рисунок 3.5).

Рисунок 3.5 – Інтерфейс сторінки тренування

Виділимо основні блоки користувацького інтерфейсу. Першим блоком є хедер сайту (рисунок 3.6), на якому сліва зображено логотип факультету ФКІТ та основна суть даного сайту або, можна сказати, його назва – Biomedical Image Recognition. Справа хедер містить навігацію по сайту, яка складається з двох маршрутів: тренування та прогнозування.

Рисунок 3.6 – Хедер сайту

React JS дозволяє розбити сторінку на прості модулі, які динамічно грузяться залежно від введеного URL і можуть бути повторно використані в інших частинах сайту. В цілому, це означає, що веб-додаток складається з окремих незалежних компонентів, які, в свою чергу, складаються з компонентів поменше, і так далі. У чому користь? Можливість розділити додаток на окремі модулі, в перспективі, добре відбивається і на розробці, і на продукті, в цілому.

При переході по URL localhost:3000/trainer або localhost:3000/predictor завантажуються відповідні компоненти Trainer.js та Predicter.js (рисунок 3.7).

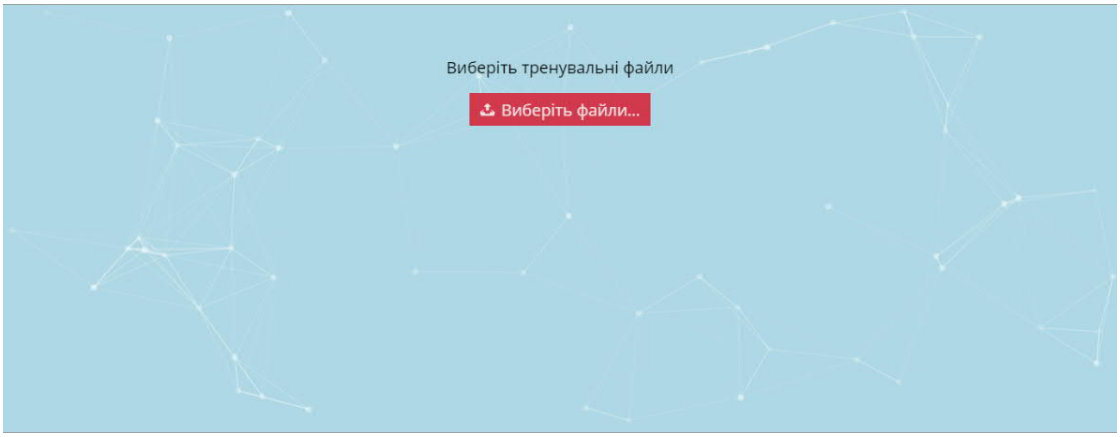


Рисунок 3.7 – Контент сайту

Нажавши на кнопку «Виберіть файли» побачимо стандартний інтерфейс браузеру Google Chrome для вибору файлів у файловій системі (рисунок 3.8).

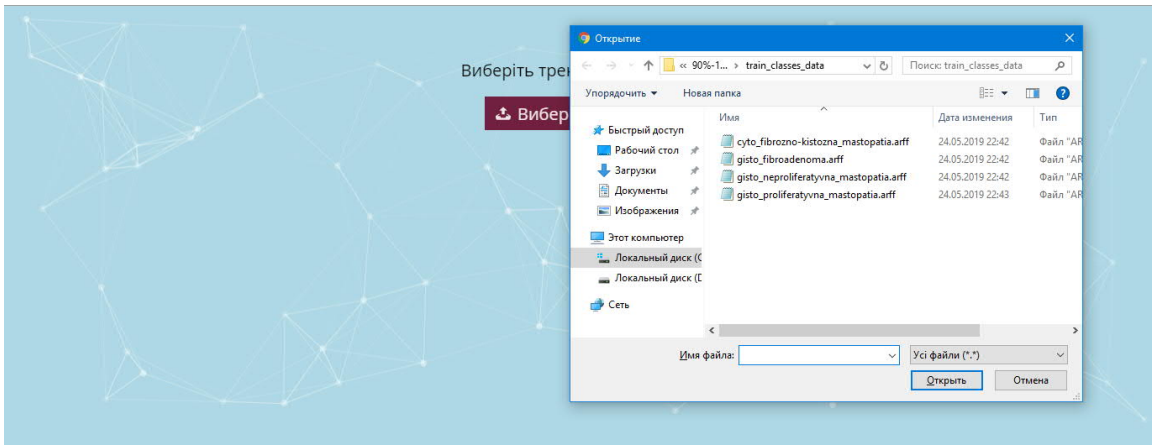


Рисунок 3.8 – Вибірка файлів при нажиманні відповідної кнопки

Останнім елементом інтерфейсу є футер (рисунок 3.9). В даному проекті він відображає ім’я та прізвище розробника та дату розробки.

Рисунок 3.9 – Футер сайту

Тепер перевіряємо функціонал. Будучи на сторінці тренування нажимаємо на кнопку «Виберіть файли», потім виділяємо файли з навчальними даними, після чого нажимаємо на кнопку «Відкрити». Після нажимання бачимо що кнопка вибору файлів пропадає, а на її місці появляються назви вибраних файлів, випадаюче меню для вибору методу тренування та дві кнопки. Виконуємо для початку класифікацію за наївним баєсівським методом, вибравши його у випадаючому меню, після чого нажимаємо кнопку «Тренувати» (рисунок 3.10).

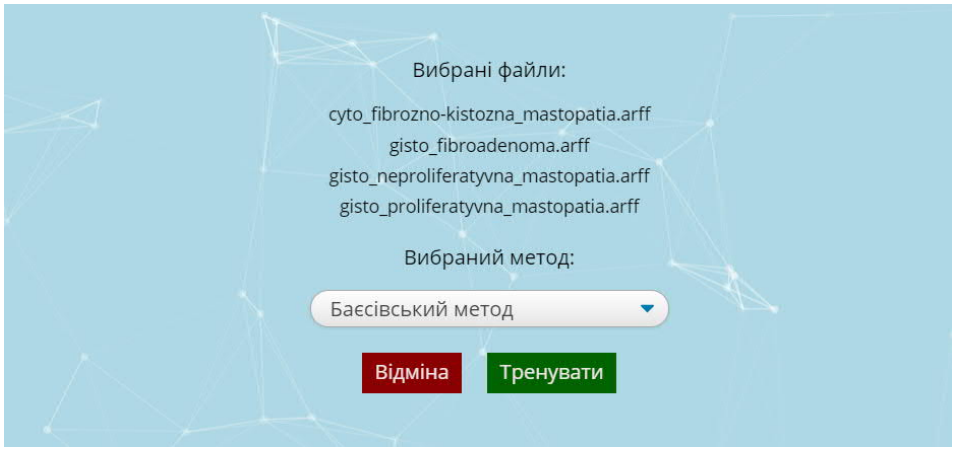


Рисунок 3.10 – Інтерфейс додатку після вибору файлів для тренування

Нажавши на кнопку з фронтенду проекту на бекенд відправиться POST HTTP запит. Даний тип запиту є найпоширенішим в клієнт-серверних додатках і використовується для відправки даних на сервер з подальшим їх збереженням в базі даних (файл data.txt в нашому проекті).

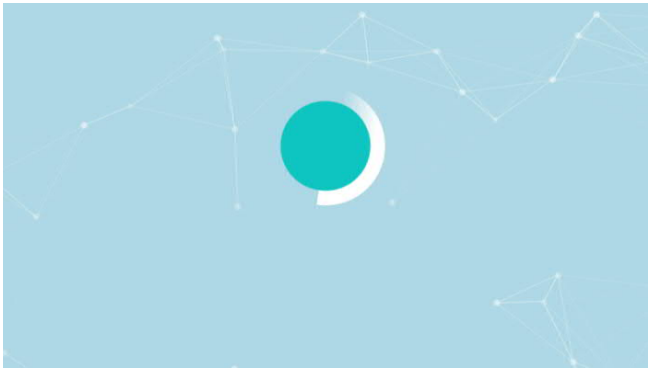


Рисунок 3.11 – Індикатор завантаження сторінки

Щоб для користувача було зрозуміло що на сайті не просто так нічого не відбувається, а відправляється HTTP запит було створено презентаційниий React компонент – індикатор завантаження сторінки (див. рисунок 3.11).

Через 1-2 секунди, залежно звичайно ж від швидкості інтернету отримаємо відповідь від сервера, яка буде виведена в консолі браузера, відкрити яку можна нажавши f12 на клавіатурі і перейшовши в панелі що появиться після нажимання у вкладку «Консоль» (рисунок 3.12) або просто нажавши комбінацію клавіш Ctrl+Shift+J, що зробить те ж саме.

Рисунок 3.12 – Браузерна консоль розробника з відповіддю з сервера

Інформація, що є для нас цікавою тут це status і його значення 200, що інформує нас про успішну обробку запиту. Ми можемо це перевірити відкривши файл data.txt в корінній папці серверної частини проекту (рисунок 3.13). В перспективі ці дані можуть зберігатися в базі даних, наприклад MySQL, будучи з'єднаними з користувачем який виконав тренування моделі. Щоб їх отримати ми б використовували інший HTTP запит – GET, після чого ці дані містилися в полі data на ристунку 3.12 і в подальшому були виведені на сайті.

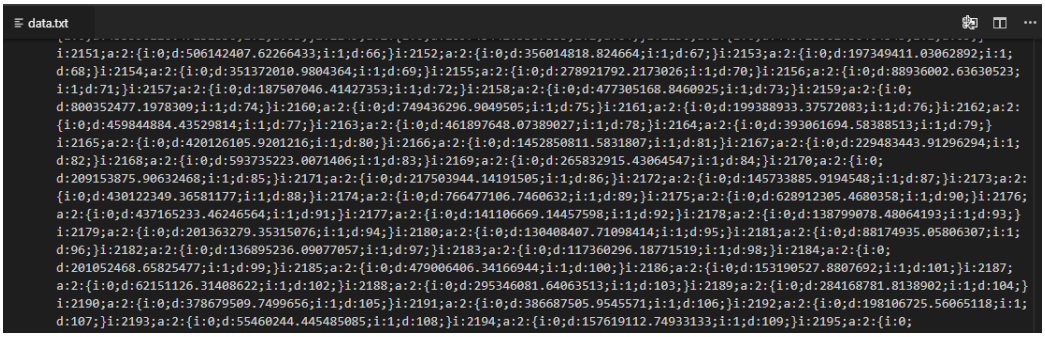


Рисунок 3.13 – Вміст файлу data.txt після тренування

Файл заповнився деякими даними, хоча до того був пустий. Реалізований функціонал для тренування моделі працює і на фронтенді і на бекенді.

Після отримання відповіді маршрутизація React.js автоматично переадресує нас на сторінку прогнозування, де ми зможемо перевірити роботу наївного баєсівського класифікатора. Можна звичайно ж явно перейти на дану сторінку нажавши по посиланню з назвою «Прогнозувати» в хедері сайту (рисунок 3.14).

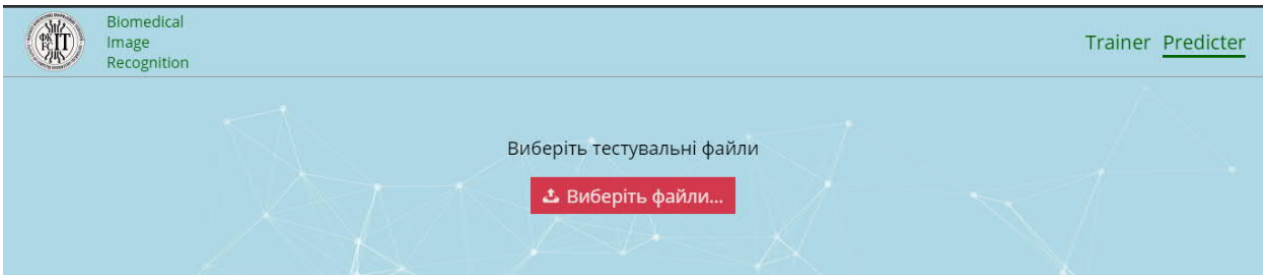


Рисунок 3.14 – Інтерфейс сторінки прогнозування

Тепер нажимаємо знову на кнопку «Виберіть файли», але цього разу вибираємо файли з тестової вибірки та нажимаємо «Відкрити». Побачимо знову ж таки, назви вибраних тестових файлів та дві кнопки. Нажимаємо на кнопку «Тестувати» (рисунок 3.15).

Рисунок 3.15 – Інтерфейс додатку після вибору файлів для прогнозування

Після нажимання відправиться той же POST HTTP запит, що включаниме вибрані нами файли тестової вибірки. При отриманні даного запиту, сервер прочитає, раніше записані дані файлу data.txt і виконавши метод predict бібліотеки PHP-ML відправить відсоток правильно відгаданої класифікатором, кількості записів тестової вибірки, і саме з такими даними інтерфейс буде зображений на сайті (рисунок 3.16).

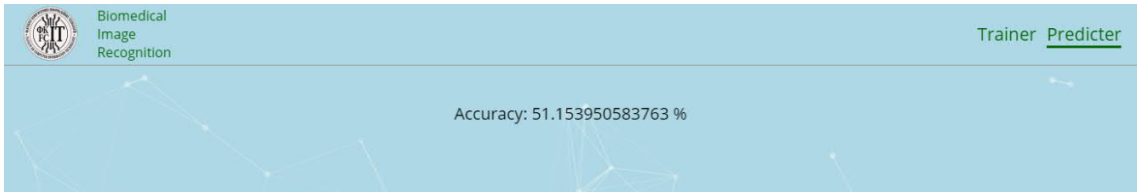


Рисунок 3.16 – Результат тестування наївного баєсівського класифікатора

Як бачимо, результат правильно відгаданої кількості файлів тестової вибірки складає приблизно 51%, що, звичайно ж є не найкращим результатом класифікації, але це було досить очікувано, оскільки наївний баєсівський класифікатор вважається одним з найнеточніших і, до того ж, не може результат класифікації бути великим якщо час прогнозування складає лише 1-2 секунди при використанні комп’ютера середньої потужності – на це потрібно набагато більше часу.

Код серверної частини можна побачити в додатку А.

3.3 Порівняльний аналіз роботи алгоритмів класифікації

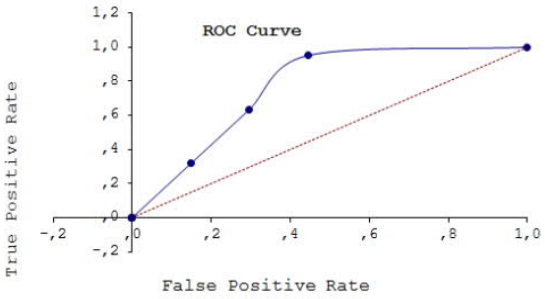
Ми живемо в часи вибухового зростання обсягів даних, що генеруються і споживаються людством. Практично в кожному з розроблюваних сьогодні додатків дані або використовуються десь всередині них, або візуалізуються.

Іноді може статися так, що найцінніше і цікаве, що може дати додаток користувачеві – це певні дані. Однак якщо уявити їх у вигляді чогось на кшталт

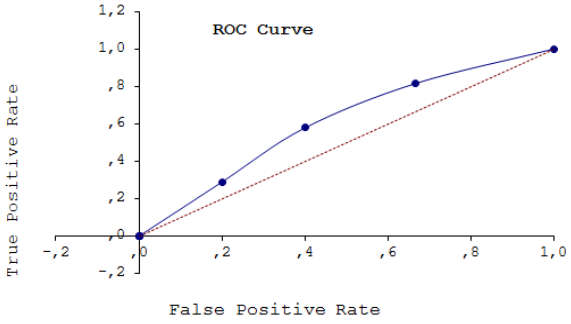
списку або таблиці, робота з такими даними, швидше за все, виявиться стомлюючою. Крім того, якщо даних багато, бачачи лише їх найпростіше уявлення, користувач зіткнеться з труднощами, що стосуються їх аналізу і прийняття на їх основі будь-яких рішень.

Результати наївного баєсівського класифікатора вже були висвітлені в попередньому підрозділі, але для справді хорошого наочного бачення результатів усіх трьох класифікаторів в даному підрозділі буде виконано порівняльний аналіз точності їх прогнозування, коригуючи відношення даних навчальної та тестової вибірок даних та кількість параметрів запису.

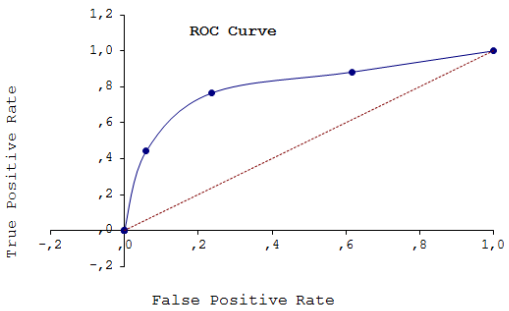
Результати класифікації цитологічних зображень за допомогою методу опорних векторів наведено на рисунку 3.17.



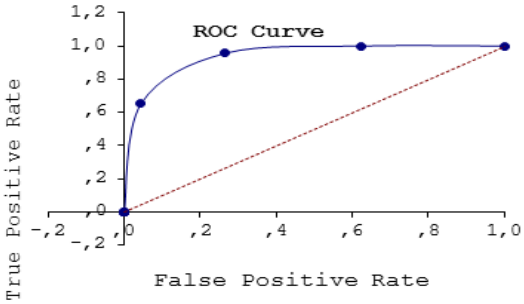
2 компоненти (AUC = 0.75)



3 компоненти (AUC = 0.6)



5 компонент (AUC = 0.79)



8 компонент (AUC = 0.91)

Рисунок 3.17 – Результати класифікації цитологічних зображень за допомогою методу опорних векторів

Приклад результату редукції даних наведено на рисунку 3.18.

0.345305783955408	0.28231351963330226
0.23754232296719177	-0.19699235840744475
-0.15595758039923657	0.5209583189516428
0.138344421491857	-0.2940285041463182
0.38782041105696563	0.028018462757987632
0.2380498752931337	0.3761418140351855
0.13213200447502207	0.10010277623690536
-0.29279723631775184	0.0237919299521784
-0.2870365342370022	0.475251202907777
-0.15881269491233355	0.18615939569286183
-0.42442284288971294	-0.24021738985944838
0.04123225652111709	-0.10944047687202607
0.05579025407436974	0.09434035585856651
0.34794452670390824	0.15200788757568442

Рисунок 3.18 – Дані після редукції (2 головні компоненти)

На рисунку 3.19 наведено зведений показник якості класифікації за допомогою методу опорних векторів на основі критерію AUC.

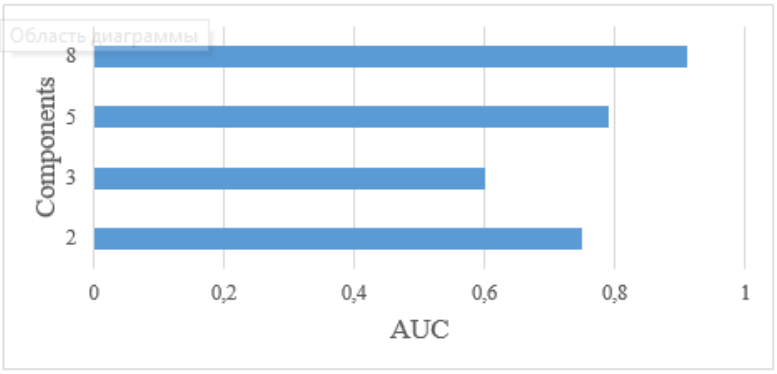
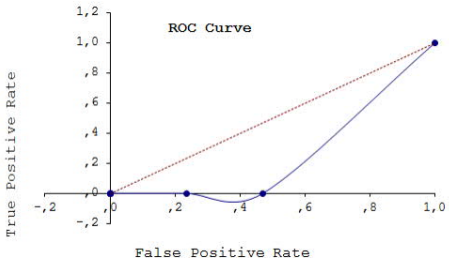


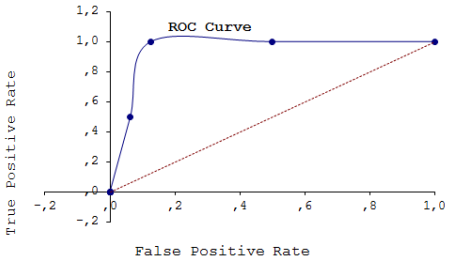
Рисунок 3.19 – Зведений показник якості класифікації за допомогою методу опорних векторів

Виходячи з результатів, наведених на рисунку 3.20 можна зробити висновок, що при використанні 8 компонент досягається найвища точність класифікації.

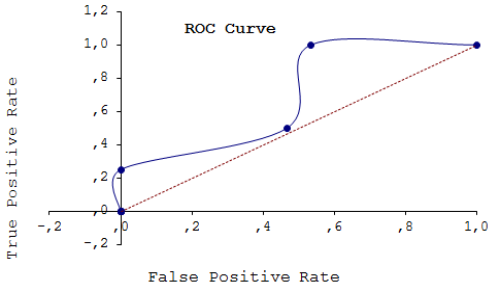
Результати класифікації за допомогою одношарового перцептрону наведено на рисунку 3.20.



2 компоненти (AUC = 0.27)



3 компоненти (AUC = 0.93)



5 компонент (AUC = 0.69)

Рисунок 3.20 – Результати класифікації за допомогою одношарового перцептрону

В результаті аналізу можна зробити висновок, що за даними показника площі під ROC кривою найкраща якість класифікації отримується при використанні 3 компонент.

Результати класифікації цитологічних зображень на основі багатокласової логістичної регресії наведено на рисунку 3.21.

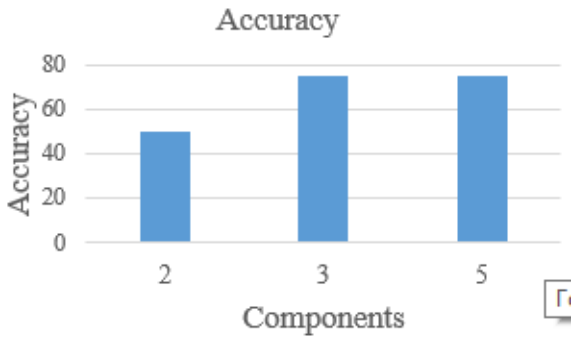


Рисунок 3.21 – Результати класифікації цитологічних зображень на основі багатокласової логістичної регресії

В результаті досліджень можна зробити висновок, що найкращий результат досягається при застосуванні 3 і 5 компонент та складає 75%.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ

4.1 Розрахунок витрат на виконання проектного рішення

4.1.1 Розрахунок витрат на розробку програмного забезпечення

Витрати на розробку і впровадження програмних засобів (*K*) рахуються за формулою 4.1.

$$K = K_1 + K_2 \tag{4.1}$$

де K_1 – витрати на розробку програмних засобів, грн;

K_2 – витрати на відлагодження і дослідну експлуатацію програми рішення задачі на комп'ютері, грн.

Витрати на розробку програмних засобів включають:

– витрати на оплату праці розробників ($B_{ОП}$);

– витрати на відрахування у спеціальні державні фонди ($B_ф$);

– витрати на покупні вироби ($П_В$);

– витрати на придбання спецобладнання для проведення експериментальних робіт ($O_б$);

– накладні витрати (*H*);

– інші витрати ($I_В$).

4.1.2 Розрахунок витрат на матеріали та комплектуючі

Матеріальні витрати – це вартість витрачених матеріалів, малоцінних та швидкозношуваних предметів на виробництво продукції, робіт або послуг, а також матеріалів і МШП, витрачених на адміністративні, збутові та інші потреби підприємства.

Загальна сума витрат на матеріальні ресурси (B_M) визначається за формулою 4.2.

$$B_M = \sum_{i=1}^{n} K_i \cdot Ц_i \,, \qquad (4.2)$$

де K_i – витрата *i*-го типу матеріалу, натуральні одиниці вимірювання;

$Ц_i$ – ціна за одиницю *i*-го типу матеріалу, грн;

i – тип матеріального ресурсу;

n – кількість типів матеріальних ресурсів.

Звідси, витрати на матеріальні ресурси дорівнюватимуть:

$$B_M = 254.1 \text{ грн.}$$

У таблиці 4.1 наведено перелік купованих виробів і розраховані витрати на них.

Таблиця 4.1 – Розрахунок витрат на матеріали та комплектуючі

№ п/п	Найменування купованих виробів	Одиниця виміру	Ціна, грн	Кількість куплених виробів	Сума, грн	Транспортні витрати (10% від суми)	Загальна сума, грн
1	Папір (формат А4)	уп	90	1	90,0	9,0	99,0
2	Ручка кулькова	шт	17	2	34	3,4	37.4
3	Олівець простий	шт	5	1	5	0,5	5.5

Продовження таблиці 4.1

4	Диски CD-R	шт	7.0	1	7.0	0,7	7.7
5	Зошит, 96 арк	шт	15.0	1	15.0	1.5	16.5
6	Тонер для принтера	уп	80.0	1	80	8	88
Разом							254.1

4.1.3 Розрахунок витрат на оплату праці та соціальні заходи

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування. Розмір ЗП обчислюється на основі трудомісткості відповідних робіт у людино-днях та середньої ЗП відповідних категорій працівників.

У розробці програмного забезпечення задіяні наступні спеціалісти – розробники, а саме: керівник проекту; студент-дипломник; консультант техніко-економічного розділу.

У таблиці 4.2 наведено вихідні дані для розрахунку витрат на оплату праці.

Таблиця 4.2 – Вихідні дані для розрахунку витрат на оплату праці

№ п/п	Посада виконавців	Місячний оклад, грн.
1	Керівник ДП, викладач	5950
2	Консультант техніко-економічного розділу, доцент	7293
3	Студент	1450

Витрати на оплату праці розробників проекту визначаються за формулою 4.3.

$$B_{ОП} = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} \cdot t_{ij} \cdot C_{ij} \,, \qquad (4.3)$$

де n_{ij} – чисельність розробників *i*-ої спеціальності *j*-го тарифного розряду;

t_{ij} – затрачений час на розробку проекту співробітником *i*-ої спеціальності *j*-го тарифного розряду;

C_{ij} – годинна ставка працівника *i*-ої спеціальності *j*-го тарифного розряду.

Середньо годинна ставка працівника розраховується за формулою 4.4.

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{РЧ_{i}}, \qquad (4.4)$$

де C_{ij} – основна місячна заробітна плата розробника *i*-ої спеціальності *j*-го тарифного розряду, грн.;

h – коефіцієнт, що визначає розмір додаткової заробітної плати (при умові наявності доплат);

$РЧ_{i}$ – місячний фонд робочого часу працівника *i*-ої спеціальності *j*-го тарифного розряду, год. (приймаємо 168 год.).

У таблиці 4.3 наведено розрахунок витрат на оплату праці.

Таблиця 4.3 – Розрахунок витрат на оплату праці

№ п/п	Посада виконавців	Час розробки, год	Погодинна заробітна плата, грн/год.	Витрати на розробку, грн
1	Керівник ДП, викладач	20	35.41	708.33

Продовження таблиці 4.3

2	Студент	120	8.63	1035.71
3	Консультант техніко-економічного розділу, доцент	10	43.41	434.1
Разом				2178.14

Величну відрахувань у спеціальні державні фонди визначають у відсотковому співвідношенні від суми основної та додаткової заробітних плат.

Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 20,5 % від суми заробітної плати:

$$B_{ф} = \frac{2178.14 \cdot 20.5}{100} = 446.51 \text{ грн.}$$

4.1.4 Витрати на використання комп'ютерної техніки

Для розробки КС використовується електрообладнання, тому необхідно розрахувати витрати на електроенергію.

Витрати на використання комп'ютерної техніки включають витрати на амортизацію комп'ютерної техніки, витрати на користування програмним забезпеченням, витрати на електроенергію, що споживається комп'ютером.

Загальна сума витрат на електроенергію розраховується за формулою 4.5.

$$B_E = \sum_{i=1}^{n} P_i \cdot k_i \cdot T_i \cdot Ц \,, \qquad (4.5)$$

де P_i – паспортна потужність *i*-го електрообладнання, кВт;

k_i – коефіцієнт використання потужності *i*-го електрообладнання;

T_i – час роботи *i*-го устаткування за весь період розробки, год;

Ц – ціна електроенергії, грн / кВт год;

і – тип електрообладнання;

n – кількість електрообладнання.

Для розробки проекту даної системи використовується один ноутбук потужністю *P* = 0,5 кВт, який за весь період розробки працює 100 годин та друкуючий пристрій потужністю *P* = 0,37 кВт, який працює 2 години.

У таблиці 4.4 наведено проміжні розрахунки на витрату електроенергії.

Таблиця 4.4 – Проміжні розрахунки на витрату електроенергії

Найменування устаткування	Паспортна потужність, кВт	Коефіцієнт використання потужності	Час роботи обладнання для розробки, год	Ціна електро-енергії,	Сума, грн.
Ноутбук	0,5	0,98	100	0,90	44.1
Принтер	0,37	0,98	2	0,90	0,65
Разом					44.75

4.1.5 Накладні витрати

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими відсотками до витрат на оплату праці.

Середньостатистичний відсоток накладних витрат приймемо 150% від заробітної плати у формулі 4.6.

$$H_B = 1.5 \cdot B_{ОП}, \quad (4.6)$$

де H_B – накладні витрати.

Розрахунок накладних витрат для даного проекту:

$$H = \frac{2178.14 \cdot 150}{100} = 3267.21 \text{ грн.}$$

4.1.6 Обчислення інших витрати

Інші витрати є витратами, які не враховані в попередніх статтях. Вони становлять 10% від заробітної плати:

$$I = 2178.14 \cdot 0.1 = 217.81 \text{ грн.}$$

Витрати на розробку програмного забезпечення рахуються за формулою 4.7.

$$K = B_{ОП} + B_{ф} + B_{ПВ} + H + I \quad (4.7)$$

$$K_1 = 2178.14 + 446.51 + 254.1 + 3267.21 + 217.81 = 6363.67 \text{ грн.}$$

Витрати на відлагодження та дослідну експлуатацію програмного продукту визначаємо за формулою 4.8.

$$K_2 = S_{м.г.} \cdot t_{від}, \quad (4.8)$$

де $t_{від}$ – комп'ютерний час, витрачений на відлагодження і дослідну експлуатацію створеного програмного продукту, год.

Загальна кількість днів роботи на комп'ютері дорівнює 30 днів. Середній щоденний час роботи на комп'ютері – 2 години. Вартість години роботи комп'ютера дорівнює 5,2 грн. Розрахунок витрат на відлагодження та дослідну експлуатацію: $K_2 = 5.2 \cdot 60 = 312$ грн.

На основі отриманих даних складаємо кошторис витрат на розробку програмного забезпечення.

Загальні витрати ($B_{КС}$) розрахуємо за формулою 4.9.

$$B_{КС} = B_{ОП} + B_{ф} + B_{М} + B_{Е} + B_{АМ} + B_{Т} + H_{В} \quad (4.9)$$

У таблиці 4.5 наведено кошторис витрат на розробку програмного забезпечення.

Таблиця 4.5 – Кошторис витрат на розробку програмного забезпечення

№ п/п	Найменування витрат	Сума витрат, грн.
1	Витрати на оплату праці	2178,14
2	Відрахування у спеціальні державні фонди	446,51
3	Витрати на куповані вироби	254,1
4	Накладні витрати	3267,21
5	Інші витрати	217,81
6	Витрати на відлагодження і дослідну експлуатацію програмного продукту	312
Разом		6675,67

4.2 Визначення експлуатаційних витрат

Для оцінки економічної ефективності розроблюваного програмного продукту слід порівняти його з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення.

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість роботи комп'ютера (за час дії програми) визначаються за формулою 4.10.

$$E_{П} = E_{1П} + E_{2П} \tag{4.10}$$

де $E_{П}$ – одноразові експлуатаційні витрати на ПЗ (аналог), грн.;

$E_{1П}$ – вартість підготовки даних для експлуатації ПЗ (аналогу), грн.;

$E_{2П}$ – вартість роботи комп'ютера для розробки програмного забезпечення (аналогу), грн.

Річні експлуатаційні витрати $B_{ЕП}$ визначаються за формулою 4.11.

$$B_{ЕП} = E_{П} \cdot N_{П} \tag{4.11}$$

де $N_{П}$ – періодичність експлуатації ПЗ (аналогу), раз/рік.

Вартість підготовки даних для роботи на комп'ютері визначається за формулою 4.12.

$$E_{1П} = \sum_{i=1}^{n} n_i t_i c_i \,, \tag{4.12}$$

де i – категорії працівників, які приймають участь у підготовці даних (i = 1,2, ... n);

n_i – кількість працівників i-ої категорії, осіб.;

t_i – трудомісткість роботи співробітників i-ої категорії по підготовці даних, год.;

c_i – середньо годинна ставка працівника i-ої категорії з врахуванням додаткової заробітної плати, що знаходиться із співвідношення формулою 4.13.

$$c_i = \frac{c_i^0 (1+b)}{m}, \qquad (4.13)$$

де c_i^0 – основна місячна заробітна плата працівника *i*-ої категорії, грн.;

b – коефіцієнт, який враховує додаткову заробітну плату (приймемо 0,57);

m – кількість робочих годин у місяці, год.

Для роботи з даними як для поточного програмного забезпечення так і аналогу потрібен один працівник, основна місячна заробітна плата якого складає: $c^0 = 1450$

Тоді:

$$c_1 = \frac{1450 \cdot (1+0)}{168} = 8.63 \text{ грн./год.}$$

Трудомісткість підготовки даних для програмного забезпечення складає 1 год., для аналога 1,5 год.

Розрахунок витрат на підготовку даних та реалізацію програмного забезпечення на комп'ютері наведено у таблиці 4.6.

Таблиця 4.6 – Розрахунок витрат на підготовку даних та реалізацію програмного забезпечення на комп'ютері

№	Час роботи співробітників, год.	Середньогодинна заробітна плата, грн./год.	Витрати, грн.
	Проектне рішення		
1	1	8,63	8,63
	Аналог		
1	1,5	11,6	17,4

Витрати на експлуатацію комп'ютера визначається за формулою 4.14.

$$E_{2П} = t \cdot S_{МГ} \ , \qquad (4.14)$$

де t – витрати машинного часу для реалізації програмного продукту (аналогу), год.;

$S_{МГ}$ – вартість однієї години роботи комп'ютера, грн./год.

$$E_{2П} = 1 \cdot 5,2 = 5,2 \text{ грн.}; \ E_{2а} = 1,5 \cdot 5,2 = 7.8 \text{ грн.};$$

$$E_{П} = 11,6 + 5,2 = 16.8; \ E_{а} = 17,4 + 7,8 = 25,2;$$

$$B_{ЕП} = 16,8 \cdot 252 = 4233,6 \text{ грн}; \ B_{еа} = 25,2 \cdot 252 = 6350,4 \text{ грн.}$$

4.3 Розрахунок ціни споживання програмного продукту

Ціна споживання – це витрати на придбання і експлуатацію програмного продукту за весь строк його служби. Ціна споживання вираховується за формулою 4.15 тоді як ціна придбання за формулою 4.16.

$$Ц_{С(П)} = Ц_{П} + B_{(Е)NPV} \ , \qquad (4.15)$$

де $Ц_{П}$ – ціна придбання програмного продукту, грн.:

$$Ц_{П} = К(1 + \frac{П_{Р}}{100}) + К_{0} + К_{к} \qquad (4.16)$$

де $К$ – кошторисна вартість;

$П_{Р}$ – рентабельність;

$К_0$ – витрати на прив'язку та освоєння програного забезпечення на конкретному об'єкті, грн.;

$К_к$ – витрати на доукомплектування технічних засобів на об'єкті, грн.

$$Ц_{П} = 15000 \cdot (1 + 0,3) = 19500 \text{ грн.};$$

$$Ц_{ПА} = 13800 \cdot (1 + 0,3) = 17940 \text{ грн.}$$

Вартість витрат на експлуатацію програмного забезпечення (за весь час його експлуатації) вираховується з формули 4.17.

$$B_{епру} = \sum_{t=0}^{T} \frac{B_{ЕП}}{(1+R)^t}, \qquad (4.17)$$

де $B_{ЕП}$ – річні експлуатаційні витрати, грн.;

$т$ – термін служби програмного забезпечення, років;

R – річна ставка проценту банку.

$$B_{впру} = \sum_{t=1}^{5} \frac{4233,6}{1+0,5} = 14112 \text{ грн.};$$

$$B_{впру} = \sum_{t=1}^{5} \frac{6350,4}{1+0,5} = 21168 \text{ грн.}$$

Тоді ціна споживання програмного забезпечення дорівнюватиме:

$$Ц_{СП} = 19500 + 14112 = 33615 \text{ грн.}$$

Аналогічно визначається ціна споживання для аналогу:

$$Ц_{СА} = 17940 + 21188 = 39108 \text{ грн.}$$

4.4 Визначення показників економічної ефективності

Економічний ефект в сфері розробки програмного продукту береться з формули 4.18.

$$E_{ПР} = Ц_{П} - Ц_{А} \quad (4.18)$$

$$E_{ПР} = 19500 - 17940 = 1560 \text{ грн.}$$

Річний економічний ефект в сфері експлуатації береться з формули 4.19.

$$E_{КС} = B_{ЕА} - B_{ЕП} \quad (4.19)$$

$$E_{КС} = 6350,4 - 4233,6 = 2116,8 \text{ грн.}$$

Додатковий економічний ефект у сфері експлуатації береться з формули 4.20.

$$\Delta E_{КС} = \sum_{t=1}^{T} E_{КС}(1+R)^{T-t} \quad (4.20)$$

$$\Delta B_{КС} = \sum_{t=1}^{5} 2116,8 \cdot (1+0,5)^{5-t} = 27915,3 \text{ грн.}$$

Сумарний ефект складає:

$$E = E_{ПР} + \Delta E_{КС} = 1560 + 27533,3 = 29475,3 \text{ грн.}$$

У таблиці 4.7 наведено показники економічної ефективності програмного забезпечення.

Таблиця 4.7 – Показники економічної ефективності програмного забезпечення.

№	Найменування	Одиниці вимірювання	Значення показників	
			Базовий варіант	Новий варіант
1	Капітальні вкладення	грн.	–	6675,37
2	Ціна придбання	грн.	17940	19500
3	Річні експлуатаційні витрати	грн.	21168	14112
4	Ціна споживання	грн.	36615	39108
5	Економічний ефект в сфері проектування	грн.	–	1560
6	Економічний ефект в сфері експлуатації	грн.	–	2116,8
7	Додатковий ефект в сфері експлуатації	грн.	–	27915,3
8	Сумарний ефект	грн.	29475,3	

Отже, після усіх техніко-економічних розрахунків ми дійшли до висновку, що показник сумарного ефекту складає 29475,3 грн.

ВИСНОВКИ

Отже, в результаті розробки дипломного проекту можна зробити наступні висновки:

1. Проаналізовано класифікацію біомедичних зображень, що дозволило краще розібратися з предметною областю.

2. Проведено аналіз і класифікацію методів класифікації даних, а також переваги та недоліки кожного з них, що дозволило виділити основні характеристики кожного з методів та відповісти на питання у правильності вибору класифікатора з допомогою нейронної мережі для даного дипломного проекту.

3. Проведено аналіз програминих засобів класифікації даних разом з перевагами та недоліками кожного з них, що дозволило обрати підходящий програмний засіб для створення нейронної мережі на основі потребностей.

4. Проаналізовано та досліджено алгоритми класифікації зображень, що дозволило реалізувати потенціал декількох з них щоб класифікувати гістологічні та цитологічні зображення.

5. Реалізовано клієнт-серверний додаток з використанням React, PHP та PHP-ML, який дозволяє користувачу натренувати модель даних, а потім виводити результат прогнозування.

6. Виконано практичний порівняльний аналіз трьох, використаних в створеному додатку алгоритмів: нейронна мережа, наївний баєсівський класифікатор та метод опорних векторів.

7. Проведено розрахунок витрат на розробку програмного забезпечення. Здійснено порівняння з існуючим аналогом, і цим показано, що дане програмне забезпечення має переваги в порівнянні з аналогами. Згідно проведеного економічного обґрунтування дане програмне забезпечення є конкурентноздатним.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке комп'ютерна томографія і як працює: веб-сайт. URL: https://znaj.ua/zdorovya/188068-shcho-take-komp-yuterna-tomografiya-i-yak-pracyuye (дата звернення: 12.12.2018).
2. Магнитно-резонансная томография: веб-сайт. URL: http://ru.wikipedia.org/wiki/Магнитно-резонансная_томография (дата звернення: 12.12.2018).
3. В чем разница между КТ, МРТ и ПЭТ?: веб-сайт. URL: https://med.vesti.ru/articles/polezno-znat/v-chem-raznitsa-mezhdu-kt-mrt-i-pet/ (дата звернення: 12.12.2018).
4. Ультразвуковое исследование: веб-сайт. URL: https://www.who.int/diagnostic_imaging/imaging_modalities/dim_ultrasound/ru/ (дата звернення: 15.12.2018).
5. Предмет гістології: веб-сайт. URL: https://studfiles.net/preview/5258227/ (дата звернення: 15.12.2018).
6. Эхокардиография сердца: принцип действия, возможности и показания к прохождению исследования: веб-сайт. URL: https://www.kp.ru/guide/iekhokardiografija-serdtsa.html (дата звернення: 27.12. 2018).
7. Модели для предсказания класса объектов: веб-сайт. URL: https://ranalytics.github.io/data-mining/023-Models-for-Class-Prediction.html (дата звернення: 27.12.2018).
8. Нейронні мережі – шлях до глибинного навчання: веб-сайт. URL: https://codeguida.com/post/739 (дата звернення: 05.01.2019).
9. Искусственное сознание (Сети: нейронные или семантические?): веб-сайт. URL: http://alephegg.narod.ru/Survay/NeuralVersusSemantic.htm (дата звернення: 05.01.2019).

10. Свёрточная нейронная сеть: веб-сайт. URL: Режим доступу: http://ru.wikipedia.org/wiki/Свёрточная нейронная сеть (дата звернення: 10.01.2019).

11. Згорткова нейронна мережа – просте пояснення CNN та її застосування: веб-сайт. URL: https://evergreens.com.ua/ua/articles/cnn.html (дата звернення: 10.01.2019).

12. Метод K-ближайших соседей для решения задачи классификации: веб-сайт. URL: https://edu.kpfu.ru/pluginfile.php/78207/mod_resource/content/1/kNN.pdf (дата звернення: 16.01.2019).

13. Как легко понять логистическую регрессию: веб-сайт. URL: https://habr.com/ru/company/io/blog/265007/ (дата звернення: 22.01.2019).

14. Six of the Best Open Source Data Mining Tools: веб-сайт. URL: https://thenewstack.io/six-of-the-best-open-source-data-mining-tools/ (дата звернення: 27.01.2019).

15. Top 15 Best Free Data Mining Tools: веб-сайт. URL: https://www.softwaretestinghelp.com/data-mining-tools/ (дата звернення: 27.01.2019).

16. Одношаровий персептрон: веб-сайт. URL: https://studfiles.net/preview/5083084/page:2/ (дата звернення: 10.02.2019).

17. Нейросетевое моделирование: многослойный персептрон: веб-сайт. URL: http://textarchive.ru/c-2643387-p4.html (дата звернення: 15.02.2019).

18. Нейронные сети для начинающих. Часть 1: веб-сайт. URL: https://habr.com/ru/post/312450/ (дата звернення: 15.02.2019).

19. Навчання штучних нейронних мереж: веб-сайт. URL: http://opticstoday.com/katalog-statej/stati-na-ukrainskom/nejromerezhi/navchannya-shtuchnix-nejronnix-merezh.html (дата звернення: 26.02.2019).

20. Brett Lantz Machine Learning with R. Birmongham-Mumbai: Pack Publishing, 2013. 396 с (дата звернення: 03.03.2019).

21. Метод опорных векторов: веб-сайт. URL: https://ru.wikipedia.org/wiki/Метод_опорных_векторов (дата звернення: 15.03.2019).

22. Классификация данных методом опорных векторов: веб-сайт. URL: https://habr.com/ru/post/105220/ (дата звернення: 15.03.2019).

23. Методи побудови математичних функцій: веб-сайт. URL: https://mylektsii.ru/13-39085.html (дата звернення: 22.03.2019).

24. Теорема Баєса: веб-сайт. URL: https://ru.wikipedia.org/wiki/ Теорема_Баєса (дата звернення: 25.03.2019).

25. Байесовский классификатор: веб-сайт. URL http://crypto.pp.ua/2011/04/bajesovskij-klassifikator/ (дата звернення: 25.03.2019).

26. Training,_validation,_and_test_sets: веб-сайт. URL: https://en.wikipedia.org/wiki/Training,_validation,_and_test_sets (дата звернення: 27.03.2019).

27. Руководство по React: веб-сайт. URL: https://metanit.com/web/react/ (дата звернення: 10.04.2019).

28. Методичні вказівки до написання техніко-економічного розділу дипломних проектів освітньо-кваліфікаційного рівня «бакалавр» підготовки 6.050102 комп'ютерна інженерія/ І.Р. Паздрій Тернопіль: ТАНГ, 2014. 37 с.

29. Методичні рекомендації до виконання дипломного проекту з освітньо-кваліфікаційного рівня "Бакалавр" напряму підготовки 6.050102 «Комп'ютерна інженерія» фахового спрямування «Комп'ютерні системи та мережі» / О.М. Березький, Л.О.Дубчак, Г.М. Мельник, Ю.М. Батько, С.В. Івасьєв / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2016. 65с.

30. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп'ютерна інженерія» / І.В. Гураль, Л.О. Дубчак / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.